TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

VILLE YLIMANNELA

DIGITAALISTEN ALLEKIRJOITUSTEN TOTEUTUS RADIOKEMIAN LABORATORIOSSA

Diplomityö

Tarkastaja: Professori Jarmo Harju
Tarkastaja ja aihe hyväksytty Tieto- ja sähkötekniikan tiedekuntaneuvoston kokouksessa 8.4.2015

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikka
Ville Ylimannela: Digitaalisten allekirjoitusten toteutus radiokemian laboratoriossa
Diplomityö, 70 sivua, 7 liitesivua
Marraskuu 2015
Pääaine: Ohjelmistotuotanto
Tarkastaja: Professori Jarmo Harju
Avainsanat: Digitaalinen allekirjoitus, radiokemia

PET-kuvantamisen yhteydessä tutkimuskohteeseen injektoitavan radioaktiivisen merkkiaineen tuotanto on tarkkaan kontrolloitu prosessi. Ennen tuotetun merkkiaineen injektoimista vaatii merkkiaine käyttöönvapautuksen. Käyttöönvapauttaja vapauttaa merkkiaine-erän allekirjoittamalla paperisen merkkiaine-eräpöytäkirjan. Siirryttäessä paperisesta eräpöytäkirjasta sähköiseen järjestelmään, täytyy allekirjoitusten muuttua fyysisistä digitaalisiksi.

Suomen lainsäädäntö asettaa vaatimuksia digitaalisille allekirjoituksille. Laissa esitellään minimivaatimukset digitaalisen allekirjoituksen toteutukselle. Lisää vaatimuksia tulee laboratoriossa noudatettavasta GMP-standardista. GMP-standardi asettaa vaatimuksia digitaalisten allekirjoitusten käytölle sähköisissä tietojärjestelmissä.

Työssä esitellään tehty digitaalisten allekirjoitusten toteutus ja verrataan tehtyä toteutusta olemassa oleviin allekirjoitusstandardeihin, Suomen lakiin, GMP-standardin vaatimuksiin ja radiokemian laboratorion tämän hetkisiin allekirjoituksiin. Löydetyt puutteet korjataan ja korjattu toteutus esitellään. Lopussa esitellään parannusehdotukset.

Alkuperäisestä toteutuksesta löydettiin useita puutteita. Useimmat puutteet olivat pieniä, mutta joukossa oli yksi vakava puute, joka olisi estänyt allekirjoitusten vahvistamisen. Korjattu toteutus korjaa miltei kaikki löydetyt puutteet. Korjattu toteutus on teknisesti parempi ja helpommin jatkokehitettävissä.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programmer in Information Technology
Ville Ylimannela: Implementation of Digital Signatures to Radiochemistry Laboratory
Master of Science Thesis, 70 pages, 7 Appendix pages
November 2015
Major: Software engineering
Examiner: Professor Jarmo Harju
Keywords: Digital Signature, Radiopharmacy

The manufacturing of tracers used in PET imaging is a strictly controlled process. Before a tracer can be injected it must be released to use. The person in charge of the quality control releases the tracer with a handwritten signature. When moving from physical, paper-based, signatures to a computerized system the signatures must also be converted from physical to digital.

The usage of digital signatures is controlled by the Finnish law. The Finnish law sets the minimum requirements which all implementations of digital signatures must fulfill. Additional requirements are set by GMP standard which is a set of guidelines overseeing the manufacturing process of pharmaceuticals. GMP standard discusses the use of digital signatures in computerized systems.

This thesis introduces the original implementation of digital signatures in PETERP project. The old implementation is compared to existing digital signature standards, the Finnish law, the requirements set by GMP standard, and the existing physical signatures from the laboratory. Major weaknesses found during the comparison are fixed and the fixes are presented to the reader. In addition, additional improvements are presented towards the end of the thesis which can be used to improve the existing implementation.

Several weaknesses were identified in the original implementation. Majority of the weaknesses were minor, however a single major weakness was found. The major weakness would have prevented the verification of digital signatures after retrieving them from database. The fixed implementation fixes most of the weaknesses. The fixed implementation is technologically more advanced and easier to develop further.

ALKUSANAT

Vuonna 2013 Atostek Oy aloitti yhden historiansa suurimmista ohjelmistoprojekteista. Projektin tarkoitus oli luoda ERP-järjestelmä PET-kuvantamiseen ja LIMS-järjestelmä radiokemian käyttöön. Projektin aikana tehtiin digitaalisten allekirjoitusten toteutus PETERP- ja LIMS-järjestelmiin. PETERP-järjestelmä otettiin käyttöön vuoden 2015 alusta. LIMS-järjestelmän kehitys jatkuu diplomityön julkaisuhetkellä.

Allekirjoittaneen lisäksi digitaalisten allekirjoituksien alkuperäisessä toteutuksessa on ollut mukana muuta projektihenkilöstöä. Allekirjoituksia kehitettäessä alkuperäisessä toteutuksessa havaittiin puutteita. Puutteiden korjaamiselle avautui diplomityön myötä erinomainen mahdollisuus. Tietoturvasta kiinnostuneena päätin tarttua tilaisuuteen ja korjata havaitut ongelmat.

Olen erittäin tyytyväinen työn lopputulokseen. Korjattu toteutus – vaikka ei edelleenkään täydellinen – korjasi kriittisimmät ongelmat alkuperäisestä toteutuksesta. Täydellisyys on asia johon monet pyrimme, mutta aina elämän realiteetit – kuten aika, osaaminen tai budjetti – eivät anna siihen mahdollisuutta.

Tahdon kiittää työn tarkastanutta professori **Jarmo Harjua**, jolta olen saanut apua työn kirjoittamisessa. Palaute on ollut arvokasta ja auttanut täydentämään työn sisältöä merkittävästi. Lisäksi haluan kiittää työn ohjaajaa **Miika Parviota** ja palautetta antanutta **Mika Torholaa**, jotka ovat väsymättömästi jaksaneet lukea, ohjata ja kommentoida työtäni. Lopuksi kiitokset **Atostek Oy**:lle työn aiheen hyväksymisestä ja rahoituksesta.

YLIMANNELA
VILLE 14751119J

Digitally signed by
YLIMANNELA VILLE 14751119J
Date: 2015.11.24 10:18:49
+02'00'

SISÄLLYSLUETTELO

1. JOHDANTO 1
2. YMPÄRISTÖ 4
2.1 Kuvantamisen prosessi 4
2.2 Merkkiainetuotanto 6
2.2.1 Elämä ennen tietojärjestelmää 7
2.2.2 LIMS 9
3. TEORIA 11
3.1 Säädökset 11
3.1.1 GMP ja digitaaliset allekirjoitukset 12
3.1.2 Suomen lainsäädäntö digitaalisista allekirjoituksista 13
3.2 Julkisen avaimen kryptografia 15
3.2.1 Digitaalinen allekirjoitus 15
3.2.2 Allekirjoituksen prosessi 16
3.2.3 Varmenteet 18
3.3 Allekirjoitustyypit 19
3.3.1 XML-allekirjoitus 19
3.3.2 BDOC 24
3.4 Allekirjoituksen välineet 25
3.4.1 Älykortti 25
3.4.2 Kortinlukija ja kortinlukijaohjelmisto 26
3.5 WYSIWYS 26
3.5.1 Ehdotus WYSIWYS-toteutuksesta 27
3.5.2 WYSIWYS ja radiokemia 29
4. ALKUPERÄINEN TOTEUTUS 30
4.1 Teknologiapakka 30
4.1.1 Matrix Gemini LIMS 31
4.1.2 PETERP 31
4.1.3 Ohjelmistoarkkitehtuuri 32
4.1.4 Allekirjoituksen toteutus 34
4.1.5 Allekirjoitettavan tiedon sarjallistamisfunktio 36
4.1.6 Allekirjoituksen tiivistäminen, luominen ja vahvistaminen 37
4.2 Allekirjoitettava tieto 38
4.2.1 Tietomalli 38
4.3 Allekirjoitettavan datan visualisointi 43
5. TOTEUTUKSEN RIITTÄVYYDEN ARVIOINTI 47
5.1 Lainsäädännön vaatimusten täyttäminen 47
5.2 Teknisessä toteutuksessa havaitut puutteet 49
5.3 Allekirjoituksien sisältö 51
5.4 Allekirjoitettavan tiedon visualisointi 53
6. KORJATTU TOTEUTUS 54

6.1 Teknisen toteutuksen korjaus 54
6.2 Tietomallin korjaus 55
6.3 Allekirjoituksien sisällön täydentäminen 57
6.4 Muutoksien testaus 59
7. JATKOKEHITYS 61
7.1 Korjatun toteutuksen jatkokehitys 61
7.1.1 Teknisen toteutuksen jatkokehitys 61
7.1.2 Allekirjoitusten sisällön jatkokehitys 62
7.1.3 Visualisoinnin jatkokehitys 62
7.2 Puhtaalta pöydältä aloittaminen 63
7.2.1 Jatkokehitettävä tietomalli 63
7.2.2 Jatkokehitysehdotuksen allekirjoitettavan tiedon visualisointi 66
8. YHTEENVETO 67
LÄHTEET 68

LIITE A: MERKKIAINEPROTOKOLLAN ERÄPÖYTÄKIRJA

LIITE B: SQL-SKRIPTI TIETOKANTASKEEMAN PÄIVITTÄMISEEN

LIITE C: TIETOMALLIN MIGRAATION TESTAAVA OHJELMAKOODI

TERMIT JA LYHENTEET

Alkuperäinen toteutus on ennen diplomityötä tehty toteutus digitaaliseen allekirjoittamiseen. Toteutus on käytössä PETERP-ohjelmistossa.

Asiakas viittaa projektin asiakkaaseen. Asiakkaana toimii Turun PET-keskus. Ei tule sekoittaa termiin asiakasohjelmisto.

BDOC on Virossa käytettävä digitaalisen allekirjoittamisen formaatti. Formaatti on zip-pakattu hakemisto, joka noudattaa ennalta määrättyä rakennetta. Itse allekirjoitus on XML-allekirjoitus.

CA Certificate Authority on entiteetti, joka myöntään varmenteita.

CT Computed Tomography (tietokonetomografia) on röngteniin perustuva kuvausmenetelmä, jossa tutkimuskohteesta luodaan 3-ulotteinen kuva ottamalla useita kuvia.

DDOC on BDOC-formaatin edeltäjä.

ERP Enterprise Resource Planning (toiminnanohjausjärjestelmä) on organisaatioiden käyttämä tietojärjestelmä, joka integroi useita eri toimintoja, esimerkiksi tuotannonohjauksen, varastohallinnan ja projektienhallinnan.

FDG Fludeoxyglucose on PET-kuvantamisessa yleisesti käytetty merkkiaine, joka hakeutuu paljon glukoosia käyttäviin soluihin, kuten syöpäkasvaimiin.

GMP Good Manufacturing Practice (hyvät tuotantotavat) ovat menettelytavat ja järjestelyt joilla varmistetaan että tuotettavat lääkkeet ovat tasalaatuisia ja täyttävät niille asetetut vaatimukset.

HTML Hyper Text Markup Language on kuvauskieli jota käytetään mm. verkkosivuilla.

IIS Internet Information Services on Microsoftin palvelinohjelmisto. Sitä voi käyttää WWW-sovelluspalvelimen ylläpitoon.

Korjattu toteutus on diplomityön aikana tehty toteutus digitaaliseen allekirjoittamiseen. Korjattu toteutus pohjautuu *alkuperäiseen toteutukseen* ja korjattu toteutus otetaan diplomityön aikana käyttöön tuotantoympäristössä.

LIMS Laboratory Information Management System on laboratorioissa käytettävä tietojärjestelmä.

MD5 Message Digest 5 on tiivistealgoritmi, joka tuottaa tuloksenaan 128-bittisen tiivisteen.

MRI Magnetic Resonance Imaging (magneettikuvaus) on kuvantamismenetelmä, joka perustuu magneettiseen resonanssiin.

PET Positron Emission Tomography (positronemissiotomografia) on lääketieteellistä kuvantamista, jossa kuvauksen kohteeseen injektoidaan radioaktiivista merkkiainetta.

PETERP on PET-kuvantamiseen suunniteltu toiminnanohjausjärjestelmä (ERP).

PETO on PETERP-ohjelmiston edeltäjä.

Projekti viittaa tämän työn kontekstissa PETERP-projektiin. Projektin tarkoituksena on kehittää PETERP-ohjelmisto ja siihen liittyvä LIMS-ohjelmisto.

Radionuklidi on radioaktiivinen isotooppi jota käytetään radiolääketieteessä.

RK Radiokemia tutkii ja tuottaa radioaktiivisten aineita.

RPC Remote Prodedure Call eli function etäkutsu on prosessien välistä kommunikointia, joka mahdollistaa keskustelun palvelimen ja asiakasohjelmiston välillä.

RSA on laajasti käytössä oleva julkisen avaimen salausalgoritmi.

SHA1 Secure Hash Algorithm on tiivistealgoritmi, joka tuottaa tuloksenaan 160-bittisen tiivisteen.

SOAP (Simple Object Access Protocol) on protokolla, joka mahdollista funktioiden etäkutsun, eli mahdollistaa palvelinkutsut asiakasohjelmistosta.

SQL Structured Query Language on kyselykieli, jolla voi tehdä hakuja tietokantaan.

URI Uniform Resource Identifier on merkkijono, joka yksilöi resurssin.

UUID Universally unique identifier on sattumanvaraisesti luotu uniikki tunniste.

W3C World Wide Web Consortium on kansainvälinen organisaatio, joka kehittää WWW:n standardeja.

WCF Windows Communication Foundation on osa NET-kirjastoa. WCF:ia käytetään esimerkiksi WWW-sovelluspalveluiden luomiseen.

WWW-sovelluspalvelu (Web service) WWW-pohjainen ohjelmointirajapinta, jonka palvelin tarjoaa asiakasohjelmistoille, usein HTTP-protokollan yli.

XML Extended Markup Language on W3C:n kehittämä rakenteellinen kuvauskieli.

1. JOHDANTO

Aloittaessani opiskeluni Tampereella vuonna 2007 oli kaupassa asioidessa normaalia, että maksaminen tapahtui pankkikortin magneettiraidalla ja allekirjoituksella. Allekirjoitus tapahtui tietenkin mustekynällä kuitille, joka jäi kauppiaalle. Vaikka useimmat maksukortit nykyään sisältävät vielä magneettiraidan, tapahtuu maksaminen useimmiten käyttämällä sirua ja käyttäjän PIN-koodia. Uudempaa tapaa pidetäänkin usein turvallisempana ja helpompana.

Sama trendi, eli muutos tavallisesta kynällä ja paperilla tehtävästä allekirjoituksesta kortti ja PIN -yhdistelmään, on tapahtumassa myös finanssimaailman ulkopuolella. Eräs aktiivinen osapuoli tässä siirtymässä on terveydenhuoltosektori. Asioidessani työterveydessä jokunen kuukausi taaksepäin, huomasin että lääkärillä oli kortti ja kortinlukija kiinni tietokoneessaan. Lopuksi lääkäri tarjosi minulle mahdollisuutta ottaa saamani resepti sähköisenä tai paperilla.

Molemmat reseptit käyvät apteekissa, mutta miten sähköiseen reseptiin voi luottaa? Fyysisessä paperilla olevassa reseptissä on lääkärin allekirjoitus, jonka avulla apteekki voi varmistua dokumentin aitoudesta. Miten tämä luottamus saadaan aikaan lääkepyynnön tullessa ulkoisesta tietojärjestelmästä? Tässä tulevat avuksi digitaaliset allekirjoitukset. Lääkäri voi allekirjoittaa digitaalisen dokumentin sirukortilla ja siihen liittyvällä PIN-koodilla. Tämän kaltainen kortilla tehty kryptografiaa käyttävä allekirjoitus on miltei mahdoton väärentää. Vuonna 2003 voimaan tulleen lain mukaan molemmat allekirjoitukset, fyysinen ja digitaalinen, ovat yhtä päteviä.

Digitaalisen allekirjoituksen vaikeasta väärennettävyydestä ja helppoudesta huolimatta ovat tavalliset allekirjoitukset kuitenkin laajasti käytössä terveydenhuoltosektorilla. Yksi merkittävä syy tähän on, että kaikesta tiedosta ei vielä ole digitaalista dokumentaatiota, jota kortilla voisi allekirjoittaa. Terveydenhuollon työntekijöiden käytössä ei ole aina tietojärjestelmää, vaan arkea pyöritetään kynällä ja paperilla. Eräs tällainen paikka on radiokemian laboratorio Turussa, jossa valmistetaan radioaktiivisia merkkiaineita PET-kuvantamista varten.

Radiokemian laboratoriossa asiat dokumentoidaan paperille. Jokaisella tuotettavalla merkkiaine-erällä on merkkiainepöytäkirja, joka tulostetaan paperille ennen tuotannon aloitusta. Tämä pöytäkirja seuraa erää kunnes erä on valmis. Merkkiainevalmistus on jaettu erilaisiin vaiheisiin. Näitä ovat esimerkiksi säteilytys, synteesi, laatukontrolli ja laadun vahvistus. Koska valmis merkkiaine tullaan injektoimaan ihmiseen, on valmistus tarkkaan kontrolloitu prosessi. Laboratoriossa pitää noudattaa erilaisia säädöksiä, standardeja ja lakeja. Tämä johtaa siihen, että merkkiaine-eräkirjan täyttäjän tarvitsee vahvistaa allekirjoituksellaan lisäämiensä tietojen paikkansapitävyys.

Tammikuussa 2013 aloitettiin projekti, jonka osana radiokemian laboratorion tuotantoprosessi tullaan siirtämään paperilta tietojärjestelmään. Tämän johdosta myös allekirjoitukset tulevat muuttumaan fyysisistä digitaalisiksi. Digitaalisen allekirjoituksen suurimmat haasteet eivät ole teknisiä, sillä modernit kehitysympäristöt tarjoavat paljon valmiita kirjastoja ja esimerkkejä näiden käytöstä. Näiden ansiosta digitaalinen allekirjoitus on teknisesti helppo asia toteuttaa. Tarvitaan vaan sarja bittejä jotka voimme allekirjoittaa valitsemallemme algoritmeilla.

Tästä pääsemmekin allekirjoittamisen problematiikkaan sisälle. Mitä allekirjoitettava tieto oikeasti on? Monessa dokumenttiformaatissa on mahdollisuus lisätä digitaalisia allekirjoituksia. Näin voi tehdä esimerkiksi Microsoft Word dokumenteille tai PDF-tiedostoille. Edellä mainitut sopivat kuitenkin huonosti tietojärjestelmiin, sillä useimmissa tietojärjestelmissä kaikki tieto päätyy loppujen lopuksi relaatiotietokantaan. Näin tapahtuu myös radiokemian laboratorion käyttöön toteutettavassa tietojärjestelmässä. Allekirjoitettava tieto sijaitsee tietokannassa, ja usein allekirjoitettava tieto on lisäksi tallessa useassa eri tietokantataulussa. Ensimmäinen ongelma on tietokannassa sijaitsevan tiedon muuntaminen sellaiseen formaattiin, jonka voi allekirjoittaa.

Eräs ratkaisu ongelmaan on tehdä tietokannassa olevasta tiedosta dokumentti, joka sitten voidaan allekirjoittaa digitaalisesti. Hyvä formaatti tällaiselle dokumentille on XML, jossa pystytään esittämään rakenteista tietoa hyvin. XML on myös laajasti tuettu formaatti, joten kirjastoja sen luomiseksi on saatavilla hyvin.

Toinen ongelma on allekirjoitettavan datan visualisointi käyttäjälle. Termiä WYSIWYS (What You See Is What You Sign) käytetään usein ongelmasta puhuttaessa. Allekirjoitettava tieto on usein monimutkaista, mutta esimerkiksi paperia allekirjoittaessaan käyttäjä näkee selvästi mitä hän on allekirjoittamassa. Digitaalisissa dokumenteissa tilanne on erilainen, varsinkin allekirjoitettavan datan ollessa hyvin vaikeaselkoista, kuten tietokannan rivejä. PDF-dokumenteissa tämä on ratkaistu siten, että allekirjoitukset näkyvät käyttäjälle. Mutta onko järkevää näyttää käyttäjälle monimutkaisia, hankalasti luettavissa olevia, monessa tapauksessa erittäin pitkiä, XML-dokumentteja? Jos ei, niin miten tässä pitäisi toimia? Voiko allekirjoittajaa pitää vastuussa allekirjoittamastaan datasta ellei sitä ole hänelle oikealla tavalla visualisoitu, eli hän ei välttämättä ole täysin tiedostanut allekirjoittamaansa tietoa?

Allekirjoituksia tehtäessä tulee huomioida useita erilaisia lakeja ja säädöksiä, joista osa tulee Suomen lainsäädännöstä ja osa EU:n tasolta. On tärkeää, että tuotantoon menevä tietojärjestelmä on kaikkien säädöksien mukainen. Tulee myös varmistua, että itse allekirjoitettava tieto on oikeaa. Kuten jo aikaisemmin mainittiin, merkkiaineiden valmistusprosessi koostuu useasta eri vaiheesta. Jokaisessa vaiheessa ei kuitenkaan allekirjoiteta koko valmistuspöytäkirjaa, vaan pelkästään se osa, jonka allekirjoittaja on itse kirjannut. Koko dokumentti allekirjoitetaan vasta lopuksi, merkkiaineen saadessa käyttöönvapautuksen.

Nämä ovat kysymyksiä ja asioita, joita diplomityö tulee tutkimaan. Tarkoitus on tarkastella jo tehtyä alkuperäistä toteutusta kriittisesti ja etsiä siitä puutteita ja heikkouksia. Löydettyjen puutteiden pohjalta allekirjoituksesta tehdään korjattu toteutus.

Työ alkaa tutustuttamalla lukija aihepiiriin. Luvussa 2 käsitellään radiokemian ja PET-kuvantamisen prosesseja ja esitellään tarvittavat lähtötiedot kokonaisuuden ymmärtämiseksi. Luku 3 käsittelee julkisen avaimen kryptografiaa ja digitaalista allekirjoittamista.

Luku 4 esittelee olemassa olevan *alkuperäisen toteutuksen*. Luku käy läpi käytetyt teknologiat ja selittää lukijalle miten digitaaliset allekirjoitukset ovat projektissa toteutettu. Luvussa 5 tätä toteutusta arvioidaan luvun 3 teorian pohjalta. Toteutuksesta löydetyt ongelmat ja heikkoudet priorisoidaan. Luvussa 6 esitellään allekirjoituksesta *korjattu toteutus*. Luku 7 esittää allekirjoitusarkkitehtuurin, joka voisi korvata nykyisen toteutuksen tulevissa projekteissa.

2. YMPÄRISTÖ

Turun Yliopistollisen sairaalan (TYKS) radiokemian laboratorio tuottaa radioaktiivisia merkkiaineita PET-kuvantamiseen. PET-kuva antaa tietoa kudosten toiminnasta, elimistön aineenvaihdunnasta ja lääkeaineiden käyttäytymisestä kudoksissa. Tämä eroaa perinteisestä kuvantamisesta siten, että PET-kuva kertoo miten elimistö käyttäytyy. Perinteisemmät kuvantamismenetelmät kuten CT ja MRI -kuvaukset kertovat miltä elimistö näyttää.

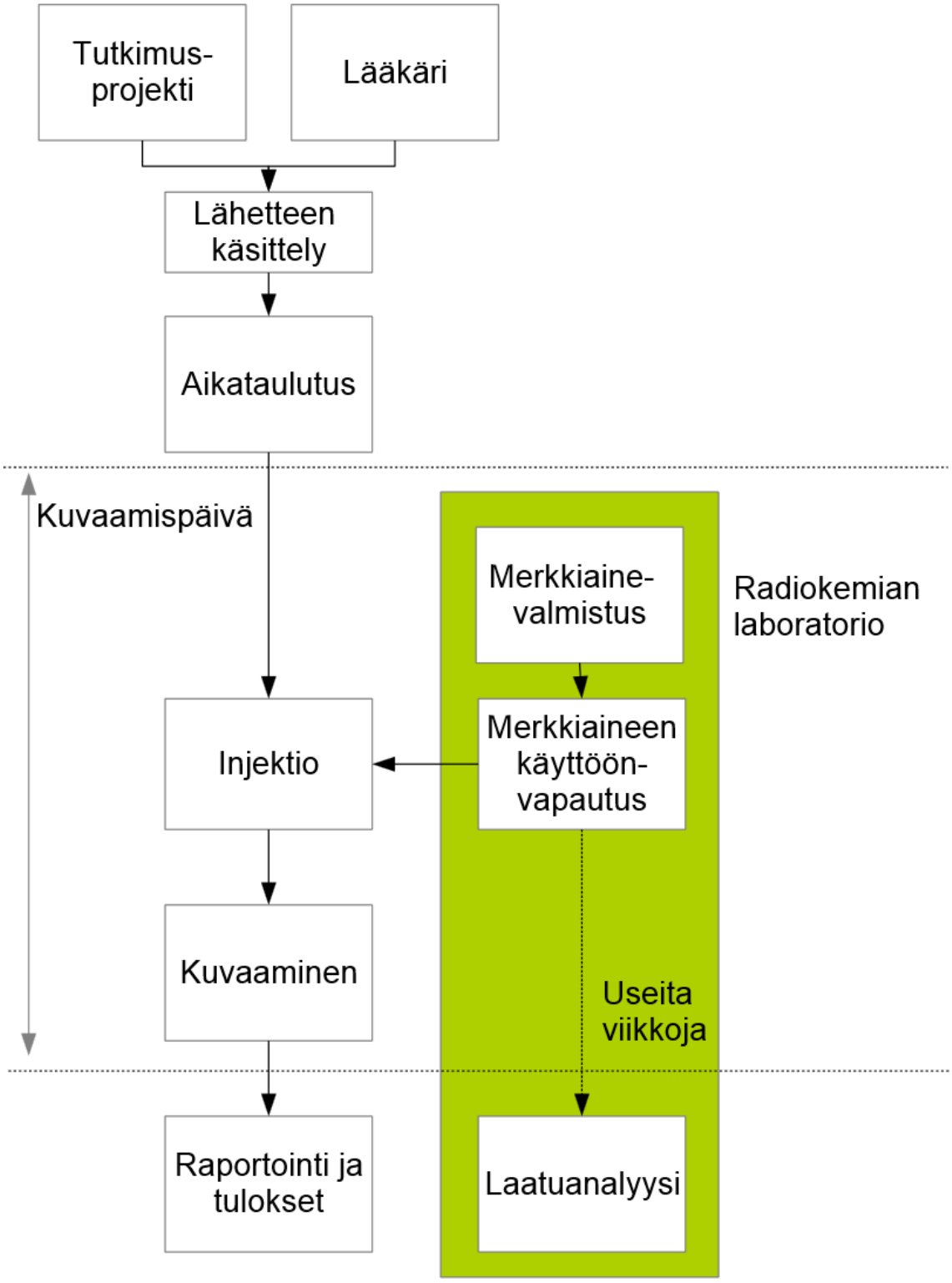
Suomessa PET-kuvantamiseen vaadittavia PET-kameroita on suurimmissa sairaaloissa. Kameran lisäksi PET-kuvantaminen vaatii radioaktiivista merkkiainetta jota injektoidaan kuvattavaan potilaaseen. Merkkiaineita valmistavia laboratorioita on Suomessa kirjoitushetkellä kaksi kappaletta. Näissä laboratorioissa valmistettavat merkkiaineet kuljetetaan laboratoriosta nopeasti kuvantamispaikalle. Ripeää toimintaa vaaditaan merkkiaineiden lyhyen käyttöiän takia.

Merkkiaine hakeutuu haluttuihin soluihin, kuten esimerkiksi syöpäsoluihin. PET-kamera näyttää mihin merkkiaine on päätynyt. Merkkiaine annetaan usein injektiona, mutta osa saatetaan antaa myös hengityksen kautta [1]. Erilaisia käytettäviä merkkiaineita on useita kymmeniä.

Merkkiaineita tuottavassa laboratoriossa toimitaan GMP-käytännön mukaisesti. GMP:n tarkoitus on taata prosessin laadunhallinta ja varmistua siitä, että prosessin tuloksena saadaan oikeanlaista, vaatimusten mukaista tuotetta. GMP-käytännöstä lisää luvussa 3.1.1.

2.1 Kuvantamisen prosessi

PET-kuvantamisen prosessi esitellään kuvassa 1. Prosessi koostuu kuudesta kokonaisuudesta: lähetteen luominen (tutkimusprojekti, lääkäri), lähetteen käsittely, kuvauksen aikataulutus, injektio, kuvaaminen ja tulosten raportointi. Prosessi etenee aina samassa järjestyksessä. Kuva 1 sisältää myös radiokemian laboratorion tehtävät, jotka esitellään tarkemmin luvussa 2.2.



Kuva 1: PET-kuvantamisen prosessi.

PET-kuvantamisen prosessi alkaa lähetteestä. Lähetteitä tulee kahdesta eri lähteestä: tutkimusprojekteista ja lääkäreiltä. Lähete päätyy sairaalan kuvantamisyksikköön. Kuvantamisyksikössä tehdään valinta, millainen kuva halutaan ottaa: millainen PET-kamera tar-

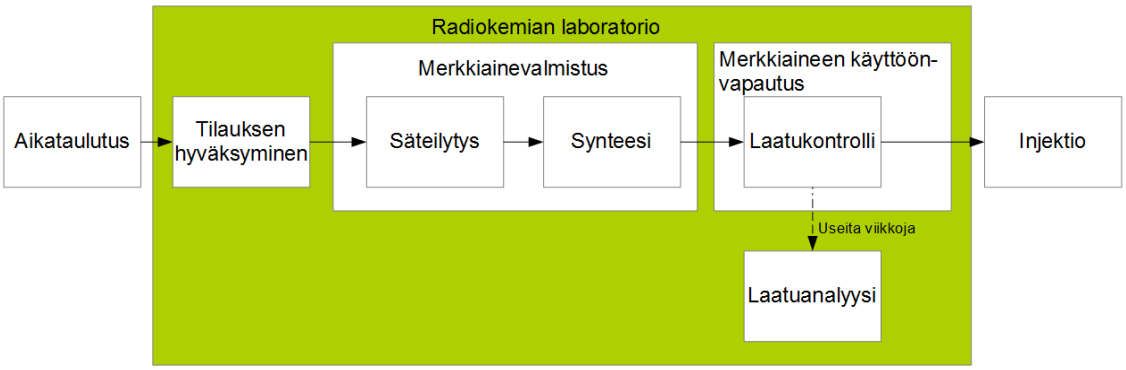
vitaan, millaista merkkiainetta kuvaukseen tarvitaan, mitä kuvataan ja millaisella kuvausprotokollalla potilas kuvataan. Kuvausprotokolla on lista vaiheita, joita kuvauksen suorittaminen vaatii, kuten esimerkiksi injektio, potilaan asettelu tai kuvaus päästä varpaisiin.

Kuvaus aikataulutetaan lähetteen sisältämän tiedon perusteella. PET-kuvantamiseen ajan saaminen kestää kauemmin kuin esimerkiksi röntgeniin pääsy, sillä kyseessä on huomattavasti monimutkaisempi prosessi. Monimutkaisuudesta johtuen aikataulutusta tehdään kuukausiksi eteenpäin. Aikataulutuksessa on otettava huomioon kuvauksen kesto, kuvauksessa tarvittava kamera (PET, PET-CT, PET-MRI) ja PET-kuvassa tarvittava merkkiaine. Useita merkkiaineita valmistetaan vain kerran kuukaudessa johtuen kysynnän rajoittuneisuudesta ja valmistukseen vaadittavien laitteiden sekä henkilökunnan saatavuudesta. Yleisiä merkkiaineita, kuten esimerkiksi FDG:ia (fluorideoksiglukoosia), valmistetaan useamman kerran viikossa.

Kuvauspäivänä radiokemian laboratorio aloittaa merkkiaineiden valmistuksen useita tunteja ennen ensimmäistä injektiota. Merkkiaineen valmistuttua se vapautetaan käyttöön. Vapautuksen jälkeen merkkiaine injektoidaan kuvattavaan henkilöön tai eläimeen. Merkkiaineen annetaan hakeutua haluttuihin soluihin jonka jälkeen kuvaus aloitetaan. PET-kuvauksen yhteydessä otetaan usein myös CT- tai MRI-kuva, jonka avulla voidaan muodostaa entistä tarkempi kuva mahdollisista ongelmakohdasta, kuten kasvaimesta [2 s. 30]. Lopuksi kuvat analysoidaan ja siitä annetaan pyytäneelle lääkärille lausunto.

2.2 Merkkiainetuotanto

Merkkiainetuotantoprosessi koostuu kuvan 2 mukaisista viidestä vaiheesta. Tuotantoprosessia kutsutaan *merkkiaineprotokollaksi*.



Kuva 2: Merkkiainetuotannon vaiheet. Prosessi etenee aikataulutuksesta injektioon kronologisessa järjestyksessä. Laatuanalyysi valmistuu vasta usean viikon jälkeen.

Prosessi alkaa vastaanottamalla tilaus. Tilaukseen liittyy tietoa, kuten mitä merkkiainetta valmistetaan, kuinka paljon aktiivisuutta tarvitaan ja milloin injektiot ovat. Tuotettavat merkkiaineet tulee tietää etukäteen, sillä merkkiaineen tuotantoon on varauduttava varaamalla tuotantoon riittävät resurssit. Resursseja ovat ihmiset, tilat, laitteet ja raaka-aineet.

Säteilytyksessä luodaan radionuklidi merkkiaineeseen käyttämällä syklotronia. Säteilytyksen lopputuloksena syntyy radioaktiivista materiaalia, joka voi olla esimerkiksi kaasuna [2, s. 126–127]. Tavallisin radionuklidi on 18F (fluori-18), jota käytetään mm. FDG-merkkiaineessa. 18F radionuklidilla on 110 minuutin puoliintumisaika, joka on pidempi kuin muilla yleisesti PET-kuvantamisessa käytetyillä radionuklideilla. [2, s. 126, s. 131]

Synteesissä yhdistetään kemikaaleja ja radionuklidi. Radionuklidien lyhyet puoliintumisajat asettavat haasteita synteesille, sillä synteesin pitää tapahtua nopeasti [2, s 131]. Synteesin lopputulos tulee allekirjoittaa synteesin tekijän toimesta. Tekijä vahvistaa allekirjoituksella että synteesi on tehty onnistuneesti. Tässä vaiheessa myös mitataan lopputuotteen radioaktiivisuus.

Synteesin tuloksena on merkkiaine. Ennen merkkiaineen injektointia pitää tarkistaa sen laatu. Tämä tapahtuu heti synteesin jälkeen laatukontrollivaiheessa, jossa merkkiaineelle tehdään laatutestejä. Testien avulla tutkitaan että merkkiaine ei sisällä mitään ylimääräisiä ja potilaalle haitallisia osia. Laatukontrolli voi tapahtua, kun merkkiainetta kuljetaan kuvantamispaikalle. Laatukontrolli tulee olla tehtynä ennen injektiota. [3, s. 11]

Mitattavia suureita ovat esimerkiksi puhtaus, ulkonäkö, pH ja steriilisyys. Kuten synteesissä, radionuklidien lyhyt puoliintumisaika asettaa laatukontrollille aikahaasteita. Tämän takia testit on tehtävä nopeasti. Tehtävät testit ovat merkkiainekohtaisia; kaikille merkkiaineille ei tehdä samoja testejä ja mitattavien suureiden hyväksymisasteikko vaihtelee merkkiaineittain. Jos merkkiaine läpäisee laatukontrollin, sille annetaan käyttöönvapautus. [3, s. 129]

Merkkiaineille tehdään myös laatuanalyysi. Ennen laatukontrollia erotetaan osa merkkiaineesta laatuanalyysiin. Laatuanalyysissä merkkiaineelle tehdään testejä, joiden tekemiseen kuluu paljon aikaa. Tästä syystä kyseisiä testejä ei ole mahdollista tehdä ennen injektiota. Laatuanalyysin tulokset valmistuvat tyypillisesti muutaman viikon kuluessa. Laatuanalyysin tulokset tulee aina allekirjoittaa, jossa laatuanalyysi joko hyväksytään tai hylätään. [3, s. 13]

2.2.1 Elämä ennen tietojärjestelmää

Ennen tietojärjestelmän käyttöönottoa merkkiaine-erän tiedot täytetään paperiseen pöytäkirjaan. Jokaiselle merkkiaineprotokollalle on oma pöytäkirjansa, jota merkkiaineen tekijät täyttävät valmistusprosessin edetessä. Liitteessä A on FTHA-tuotantoprotokollan merkkiainepöytäkirja.

Pöytäkirja

Valmistettavalle merkkiaineelle on aina valmistuspöytäkirja. Paperista pöytäkirjaa täytetään valmistusprosessin edetessä. Jokaiselle merkkiaineprotokollalle on oma pöytäkirjansa, sillä jokainen valmistusprosessi on erilainen ja tästä johtuen yleiskäyttöisen version luominen on hankalaa.

Pöytäkirja koostuu vaiheista ja askelista. Vaiheet ovat merkkiainevalmistuksen isoja kokonaisuuksia jotka esitetään kuvassa 2. Vaiheet koostuvat pienemmistä askelista. Askeleet listaavat kaikki merkkiainetuotannossa käytetyt raaka-aineet, niiden eränumerot, viimeiset käyttöpäivät ja kuinka paljon kyseistä ainetta on käytetty. Tämän lisäksi pöytäkirjassa on askelia joihin kirjataan tehtyjen testien tulokset. Esimerkiksi Laatukontrollin (Quality Control) alle merkitään pH-arvo ja radiokemikaalinen puhtaus.

Pöytäkirjassa on jokaiselle askeleelle merkattava tekijä. Tekijältä vaaditaan nimikirjaimet tai allekirjoitus. Nimikirjaimet vaaditaan jokaisessa askeleessa ja ne kirjoitetaan pöytäkirjan oikealla olevaan ”Initials”-kenttään. Pöytäkirjassa on paikka kolmelle allekirjoitukselle. Allekirjoitukset vaaditaan ”Synthesis made by”, ”Batch released by” ja ”QA approved by” -kohtiin.

Allekirjoituspisteet

Jokaiselle merkkiaine-erälle on vähintään neljä allekirjoituspistettä: tilauksen vastaanotto, synteesi, käyttöönvapautus ja laatuanalyysi. Jotkin merkkiaineet vaativat viidennen allekirjoituksen ihmiskäyttöä varten (QC-vapautus taulukossa 1). Allekirjoitusoikeudet ovat merkkiainekohtaisia. Tämä tarkoittaa, että jos henkilö saa antaa käyttöönvapautuksen merkkiaineelle FDG, ei se tarkoita että hän saa antaa käyttöönvapautuksen merkkiaineelle FTHA. Tällä hetkellä allekirjoitusoikeuksia hallitaan paperilla. Jokaiselle merkkiaineen allekirjoituspisteelle on listattu tekijät jotka saavat allekirjoitukset antaa.

Taulukossa 1 on merkkiainetuotannon allekirjoituspisteet. Allekirjoitukset esitellään järjestyksessä, jossa ne tulee antaa.

Taulukko 1: merkkiainetuotannon allekirjoituspisteet.

Allekirjoituspiste	**Selite**
Tilauksen vastaanotto	Merkkiainetilauksen saapuessa laboratorioon tilaus kuitataan vastaanotetuksi, jonka jälkeen tilaus siirtyy tuotantoon. Tämä allekirjoitus tulee jokaiselle erälle. Oikeus ei ole merkkiainekohtainen.
Synteesi	Synteesintekijä vakuuttaa merkkiaineen synteesin onnistuneen. Tämä allekirjoitus tulee jokaiselle erälle.
Käyttöönvapautus	Käyttöönvapauttaja vakuuttaa että merkkiaine on kunnossa ja vapauttaa sen käyttöön. Allekirjoituksen jälkeen merkkiaineen voi injektoida. Tämä allekirjoitus tulee jokaiselle erälle.

QC-vapautus	Osa merkkiaineista vaatii edellisen käyttöönvapautus-allekirjoituksen lisäksi toisen allekirjoituksen ennen kuin merkkiaine saadaan injektoida ihmiseen. QC-vapautusta ei tule jokaiselle erälle.
QA-hyväksyntä / QA-hylkäys	Merkkiaineille tehdään syvällisempi laatuanalyysi, joka valmistuu usean viikon päästä merkkiaineen valmistuksesta. Tämä allekirjoitus tulee jokaiselle erälle.

Laboratoriossa tarvitaan allekirjoituksia valmistusprotokollille. Valmistusprotokollia on merkkiaineille, merkkiaineiden lopputuotepulloille ja merkkiainetuotannossa käytettäville liuoksille. Jokainen käytössä oleva protokolla vaatii yhteensä neljä allekirjoitusta. Nämä allekirjoitukset esitetään taulukossa 2.

Taulukko 2: merkkiaineohjeen hyväksyntään tarvittavat allekirjoitukset.

Allekirjoitus	**Selite**
Ohjeen tekijä	Henkilö joka on tehnyt merkkiaineohjeen
Tarkistaja	Kaksi henkilöä jotka ovat tarkistaneet merkkiaineohjeen. Molemmat allekirjoittavat ohjeen.
Hyväksyjä	Henkilö joka hyväksyy lopullisen merkkiaineohjeen.

Nykyään kaikki allekirjoitukset ovat paperilla. LIMS-järjestelmän myötä on tarkoitus siirtyä digitaalisiin allekirjoituksiin, sillä myös valmistusprotokollat tulevat olemaan sähköisiä.

2.2.2 LIMS

LIMS (Laboratory Information Management System) on laboratorioihin suunniteltu tietojärjestelmä. Tietojärjestelmän on tarkoitus vähentää turhaa paperia, vähentää kirjoitus- ja merkintävirheiden määrää ja helpottaa raportointia. Projektissa toteutettavan LIMS-järjestelmän on tarkoitus digitalisoida jo olemassa oleva prosessi. Prosessiin ei tule muutoksia. Myös merkkiaineprotokollat tulevat pysymään entisellään, mutta siirtyvät paperilta tietojärjestelmään.

Projektissa on tarkoitus korvata paperi teknologialla. Tämä tarkoittaa, että järjestelmän tulee hallita seuraavat asiat: käyttäjienhallinta, digitaaliset allekirjoitukset, radiokemian

prosessin mallintaminen, raportointi, historiatietojen säilyttäminen ja kaikkeen tietoon kohdistuva muutoshistoria (Audit Trail).

3. TEORIA

Tässä luvussa käsitellään työn taustalla olevaa teoriaa. Tarkoituksena on antaa lukijalle puitteet ymmärtää mitä digitaaliset allekirjoitukset ovat ja mitä niillä halutaan saavuttaa. Luvussa 3.1 tarkastellaan erilaisia säädöksiä ja lakeja, jotka tulevat vastaan laboratorioympäristössä sekä tutkitaan Suomen lakia digitaalisten allekirjoitus kannalta.

Luku 3.2 selittää julkisen avaimen kryptografiaa ja digitaalisen allekirjoittamisen hyötyjä. Luvussa 3.3 tarkastellaan digitaaliseen allekirjoittamiseen luotuja standardeja: XML-allekirjoitukset ja XML-allekirjoituksiin pohjautuva BDOC-standardi. Luvun 3.4 tarkoituksena on avata lukijalle mitä allekirjoittamiseen käytännössä tarvitaan: älykortti, kortinlukija ja kortinlukijaohjelmisto.

Mikael Lindenin lisensiaatintutkimus [18] on hyvä lähde julkisen avaimen järjestelmän ymmärtämiseksi. Lisensiaatintutkimuksessaan Linden käy läpi julkisen avaimen kryptografian periaatteet ja avaa älykorttien toimintaa.

Lopuksi luvussa 3.5 tarkastellaan digitaalisten allekirjoitusten problematiikkaa. Digitaalisissa allekirjoituksissa tulee vastaan ongelmia, joita ei fyysisissä allekirjoituksissa ole. Kappaleessa avataan termin WYSIWYS tarkoitusta, semanttisen välimatkan merkitystä ja miten ne vaikuttavat digitaalisiin allekirjoituksiin.

3.1 Säädökset

Laboratoriossa valmistettavat merkkiaineet päätyvät injektion kautta ihmiseen. Tästä syystä prosessia säädellään GMP-standardilla. GMP-standardia käytetään elintarvikkeiden sekä lääkeaineiden valmistuksessa, johon myös merkkiaineet sisältyvät. Standardin tarkoitus on huolehtia, että prosessi tuottaa tasalaatuisia aineita ja aineet ovat turvallisia käyttää.

Laki vahvasta sähköisestä tunnistautumisesta ja sähköisistä allekirjoituksista vuodelta 2003 säätelee digitaalisten allekirjoitusten käyttöä Suomessa. Laki selkiytti digitaalisten allekirjoitusten käyttöä tietojärjestelmissä antamalla tulkinnan siitä, mitä digitaalisella allekirjoituksella tarkoitetaan. Luvussa 3.1.2 tiivistetään digitaalisen allekirjoituksen kannalta tärkeimmät lain asettamat vaatimukset. [13]

3.1.1 GMP ja digitaaliset allekirjoitukset

GMP:n käytöstä on annettu EY-asetus n:o 2023/2006 jonka mukaan tuotteiden valmistajien on huolehdittava, että tuotannossa käytetään hyviä tuotantotapoja. Hyvät tuotantotavat tarkoittavat prosessin laadunhallintaa, jonka tuloksena oikeanlaisista ja jäljitettävissä olevista raaka-aineista valmistetaan vaatimuksenmukaisia tuotteita. [15]

EU:n säädös GMP:stä määrittelee käytäntöjä koskien tietojärjestelmiä. Säädös käsittelee digitaalisia allekirjoituksia hyvin yleisellä tasolla. GMP-standardissa luetellaan seuraavat asiat (käännetty englannista) [16]:

a. Elektronisia tallenteita voidaan allekirjoittaa digitaalisesti.
b. Niillä tulee olla sama vaikutus kuin käsin kirjoitetuilla allekirjoituksilla yrityksen sisällä.
c. Allekirjoitukset pitää linkittää pysyvästi niihin liittyvään tietoon.
d. Allekirjoitusten tulee sisältää päivämäärä ja kellonaika jolloin ne on tehty.
e. Tuote-erien vapautuksessa tulisi (alkuperäisessä englanninkielisessä tekstissä käytetään sanaa ”should”) käyttää digitaalisia allekirjoituksia. Vain sallittujen henkilöiden pitäisi pystyä käyttöönvapauttamaan tuote-eriä. Järjestelmän tulee tunnistaa käyttäjät ja tallentaa käyttöönvapauttajan tunnistetiedot.

Kohta a sanoo, että allekirjoittaminen digitaalisesti on mahdollista. Kohta ei pakota digitaalisia allekirjoituksia järjestelmään, vaan niiden käyttö säilyy optionaalisena. On mahdollista käyttää tietojärjestelmää, jossa ei ole digitaalisia allekirjoituksia. Kohta ei ota kantaa mitä allekirjoitetaan tai kuka allekirjoittaa.

Kohta b on tulkittavissa siten, että kaikki dokumentit, joissa on käsin kirjoitettuja allekirjoituksia, voidaan korvata digitaalisilla allekirjoituksilla. Kohta pätee kuitenkin vain yrityksen sisällä. On mahdollista, että dokumentti on toimitettava toiselle organisaatiolle. Tässä tapauksessa digitaaliset allekirjoitukset voivat olla ongelmallisia, mikäli toinen organisaatio ei niitä hyväksy. Merkkiainetuotannon kontekstissa tämä voi tarkoittaa tilannetta jossa esimerkiksi lääkeyhtiö pyytää eräpöytäkirjaa tilaamiensa kuvausten merkkiaineesta.

Kohta c vaatii, että alkuperäinen data tulee olla linkitettynä allekirjoitukseen. Allekirjoitusta ei saa toteuttaa siten, että allekirjoitettu data voi muuttua tai kadota. Linkki allekirjoituksen ja allekirjoitetun datan tulee säilyttää pysyvästi. Allekirjoitetusta datasta tulee ottaa kopio säilytettäväksi siten, että sen muokkaus tai poistaminen ei ole mahdollista.

Kohdan d tulkinta on yksinkertainen: allekirjoitettavan datan tulee sisältää aikaleima. Tietokannan tai palvelimen generoima aikaleima ei riitä, vaan aikaleima tulee olla myös itse allekirjoituksessa.

Kohta e suosittelee digitaalisten allekirjoitusten käyttämistä merkkiaine-erien vapautuksessa. Kohdassa suositellaan, että käyttäjät tulee tunnistaa luotettavalla tavalla. GMP-standardissa suositellaan, että vain pätevä käyttäjä pystyy vapauttamaan tuote-erän. Kohta ei vaadi yhtään mitään, mutta linjaa toimintatavan, jota tietojärjestelmän olisi toivottavaa noudattaa. Kohdan tärkein anti on: vain luotettavasti tunnistetun ja pätevän käyttäjän tulee antaa vapauttaa merkkiaine-eriä. Vaikka kohta e puhuu vain tuote-erien vapautuksesta, ei se estä samojen hyvien periaatteiden soveltamista muihin merkkiainetuotannon allekirjoituspisteisiin.

3.1.2 Suomen lainsäädäntö digitaalisista allekirjoituksista

Suomessa säädettiin vuonna 2003 *laki vahvasta sähköisestä tunnistamisesta ja sähköisistä allekirjoituksista*. Lain mukaan digitaaliset allekirjoitukset ovat yhtä päteviä kuin fyysiset allekirjoitukset.

Laki vahvasta sähköisestä tunnistamisesta ja sähköisistä allekirjoituksista määrittelee termin *kehittynyt sähköinen allekirjoitus*, millä tarkoitetaan allekirjoitusta [13]:

a. joka liittyy yksiselitteisesti sen allekirjoittajaan;
b. jolla voidaan yksilöidä allekirjoittaja;
c. joka on luotu menetelmällä, jonka allekirjoittaja voi pitää yksinomaisessa valvonnassaan; ja
d. joka on liitetty muuhun sähköiseen tietoon siten, että tiedon mahdolliset muutokset voidaan havaita;

Kohdat a ja b täyttyvät, kun allekirjoittajalle luodaan oma, yksilöllinen, avainpari. Kohta a tarkoittaa, että käytettävän kryptografian on oltava riittävän vahvaa, ettei allekirjoitusta voi väärentää. Käytössä on oltava jokin hyväksi todetuista julkisen avaimen salauksista. Kohta b vaatii, että allekirjoituksessa on tieto, joka kertoo kenelle kyseinen allekirjoitus kuuluu.

Kohta c täyttyy, mikäli allekirjoitusvarmenteen yksityinen avain on saatavilla vain allekirjoittajalle. Tätä varten on olemassa älykortteja, joista enemmän luvussa 3.4.1. Tämän projektin kontekstissa käytetään VRK:n myöntämiä kortteja, jotka toteuttavat kohdat a-c.

Kohta d vaatii kahta asiaa: linkkiä tiedon ja allekirjoituksen välille sekä muutosten havaitsemisen. Jälkimmäinen on toteutettavissa käyttämällä julkisen avaimen kryptografiaa. Julkisen avaimen kryptografiaa ja digitaalisia allekirjoituksia on käsitelty tarkemmin luvussa 3.2.2. Kryptografian avulla voi tarkastaa onko dokumentti muuttunut. Mikäli dokumentti on muuttunut, ei kryptografialla saa vastausta kysymykseen ”Mikä on muuttunut?”.

Kohta d vaatii, että allekirjoitus täytyy olla liitettynä allekirjoitettuun tietoon. Laki ei kuitenkaan täsmennä miten tämä linkki on tehtävä: riittääkö että allekirjoitus on samassa tietokannassa ja tietokannassa on luotu linkki allekirjoitetun datan ja allekirjoituksen välille vai pitääkö allekirjoituksen olla samassa tiedostossa. Laki on kuitenkin tulkittavissa siten, että allekirjoituksen ei tarvitse sijaita samassa tiedostossa.

Termillä *allekirjoituksen luomisväline*, tarkoitetaan ohjelmistoja ja laitteita, joilla yhdessä luomistietojen kanssa luodaan sähköinen allekirjoitus. *Allekirjoituksen luomistiedoilla* tarkoitetaan allekirjoituksen luomisessa käytettävää ainutkertaista tietokokonaisuutta, kuten koodeja ja yksityisiä avaimia. Näitä laitteita ovat esimerkiksi kortinlukija ja siihen liittyvä kortinlukuohjelmisto. Luomisvälineen on varmistettava, että [13]:

a. allekirjoituksen luomistiedot ovat käytännössä ainutkertaisia ja että ne säilyvät luottamuksellisina;
b. allekirjoituksen luomistietoja ei voida päätellä muista tiedoista;
c. allekirjoitus on suojattu väärentämiseltä;
d. allekirjoittaja voi suojata allekirjoituksen luomistiedot muiden käytöltä; sekä
e. luomisväline ei muuta allekirjoitettavia tietoja eikä estä tietojen esittämistä allekirjoittajalle ennen allekirjoittamista.

Luomisvälineen tulkinta voi olla hieman ongelmallinen, sillä laki ei suoraan ota kantaa mikä kaikki on digitaalista allekirjoittamista. Digitaalinen allekirjoitus koostuu useasta vaiheesta: tiedon näyttäminen käyttäjälle, tiedon sarjallistaminen, tiivistäminen ja tiivisteen salaaminen yksityisellä avaimella. Kohta e kuitenkin sanoo, että luomisväline ei muuta allekirjoitettavaa tietoa. Tiedon tiivistäminen yksisuuntaisella tiivistefunktiolla luo allekirjoitettavan tiedon. Tästä voisi päätellä, että ennen tiivistämistä tapahtuvat toiminnot eivät ole allekirjoittamista. Ne ovat kuitenkin pakollisia vaiheita allekirjoitusprosessissa. Kohdan e voi tulkita siten, että allekirjoituksella tarkoitetaan prosessia jossa tiiviste lähetetään älykortille allekirjoitettavaksi ja tätä tiivistettä luomisväline ei saa muokata.

Tätä tulkintaa vahvistaa Liikenne ja viestintäministeriön julkaisu ”Turvalliset sähköisen allekirjoituksen luomisvälineet” jonka yhteenvedossa sanotaan:

> ”Yleisesti ottaen näillä kaikilla termeillä [allekirjoituksen luomisvälineellä ja luomismenetelmällä] tarkoitetaan itse fyysisen laitteen ja sen toiminnan mahdollistavien ohjelmistokomponenttien muodostamaa kokonaisuutta. Esimerkkinä mainittakoon toimikortti, sen lukijalaite, lukijalaitteen ajurit sekä esimerkiksi selainkäytön mahdollistavat ohjelmistomoduulit. ” [25]

Lainaus vahvistaa esitettyä tulkintaa, että älykortti, kortinlukija sekä näiden ajurit muodostavat kokonaisuuden jota kutsutaan allekirjoituksen luomisvälineeksi. Allekirjoituksen luomisvälineellä ei viitata koko allekirjoitusprosessissa käytettyyn ohjelmistopinoon.

3.2 Julkisen avaimen kryptografia

Julkisen avaimen kryptografia perustuu kahteen avaimeen: julkiseen ja yksityiseen avaimeen. Kyseessä on epäsymmetrinen kryptografia. Tämä tarkoittaa, että viestin salaukseen käytetään eri avainta kuin sen purkamiseen. Jos viesti salataan julkisella avaimella, voi sen avata vain yksityisellä avaimella. Epäsymmetrinen kryptografia toimii myös toiseen suuntaan: yksityisellä avaimella salatut viestit voidaan avata julkisella avaimella. Luku 3.2.2 selittää miten tämä mahdollistaa digitaaliset allekirjoitukset.

Julkisen avaimen kryptografiassa käyttäjällä on pääsy omaan yksityiseen avaimeensa. Julkinen avain on julkista tietoa ja se on osa varmennetta. Väestörekisterikeskuksen myöntämät varmenteet ovat saatavilla VRK:n verkkosivuilta. Varmenteen sisältämän julkista avainta vastaava yksityinen avain löytyy älykortin sirulta.

3.2.1 Digitaalinen allekirjoitus

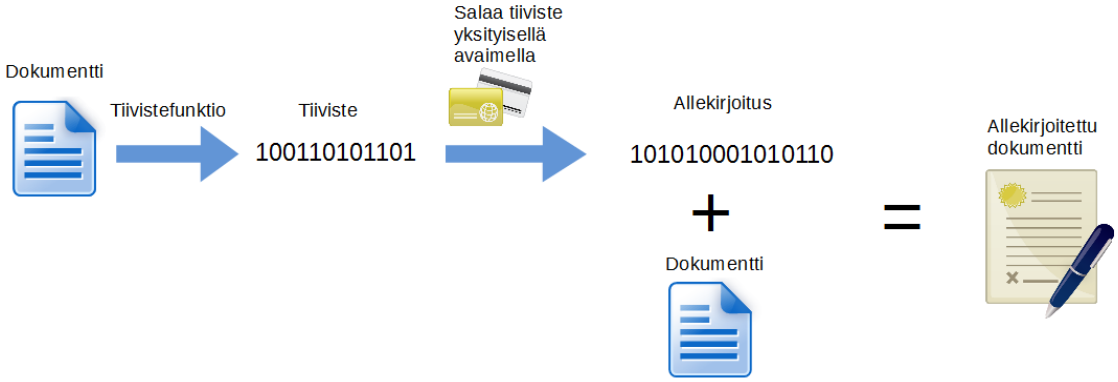
Digitaalisella allekirjoituksella käyttäjä voi varmentaa dokumentin autenttisuuden. Allekirjoitus tapahtuu yksityisellä avaimella. Digitaalinen allekirjoitus vastaa monella tavalla fyysistä kynällä paperille tehtävää allekirjoitusta. Molemmissa allekirjoituksissa on hyvät ja huonot puolensa. Digitaalisen allekirjoituksen suurimmat hyödyt ovat:

- Allekirjoittamisen nopeus: digitaalista allekirjoitusta varten dokumenttia ei tarvitse skannata paperilta ja lähettää vastaanottajalle. Lisäksi allekirjoitetun dokumentin tarkastaminen voidaan automatisoida.
- Dokumentin aitous (authenticity): digitaalisia allekirjoituksia on mahdoton väärentää ilman yksityistä avainta. Tämä mahdollistaa allekirjoittajan luotettavan tunnistamisen.
- Dokumentin eheys (integrity): kaikki muutokset huomataan. Dokumenttia ei ole mahdollista muokata ilman että allekirjoitukset muuttuvat mitättömiksi.

Merkittävä haittapuoli digitaalisissa allekirjoituksissa on erilaisten allekirjoitusstandardien ja teknologioiden määrä. Erilaiset allekirjoitusstandardit eivät ole toistensa kanssa yhteensopivia. Käyttäjä voi tarvita useita eri ohjelmia allekirjoitusten vahvistamiseen jos allekirjoituksia tulee useasta eri lähteestä.

3.2.2 Allekirjoituksen prosessi

Digitaalinen allekirjoitus tapahtuu kuvassa 3 esitetyn prosessin mukaisesti.

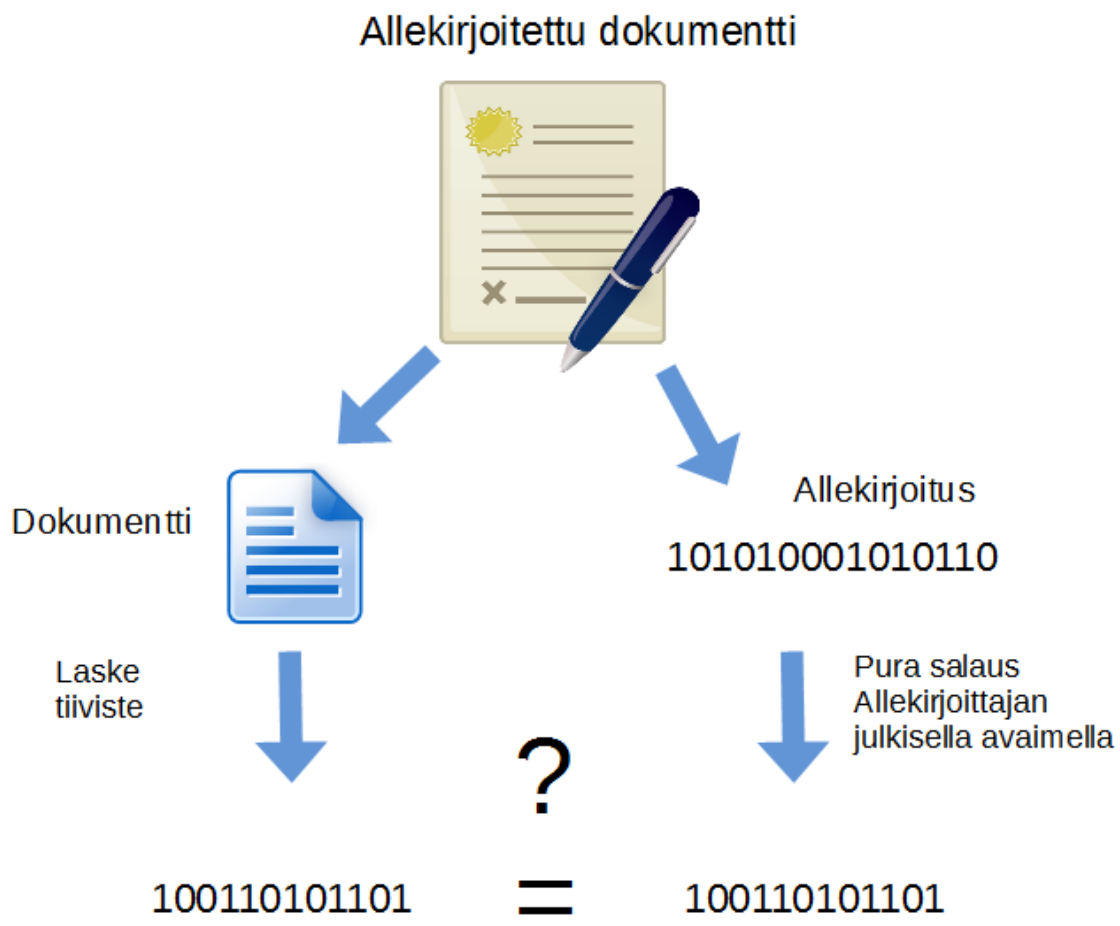


Kuva 3: Digitaalisen allekirjoituksen prosessi.

Prosessi alkaa valitsemalla allekirjoitettava data. Allekirjoitettava data voi olla esimerkiksi XML-tiedosto, pdf-dokumentti tai sähköposti. Dokumentista lasketaan tiiviste yksisuuntaisella tiivistealgoritmilla. Tiivistäminen ei ole pakollista, mutta se nopeuttaa allekirjoituksen luontia ja pienentää allekirjoituksen kokoa. Tiivisteen laskemisessa käytettäviä algoritmeja ovat esimerkiksi MD5, SHA-1 ja SHA-2. Tiiviste on yleensä vähintään 128 bittiä pitkä arvo. Vanhempia algoritmeja, kuten MD5-algoritmia, ei tulisi käyttää. MD5-algoritmista on löytynyt törmäyksiä jotka heikentävät sen luotettavuutta [26]. Myös siirtymää SHA-1-algoritmista SHA-2-algoritmiin on suositeltu [27]. Suositus koskee erityisesti nettiselainten valmistajia ja CA-toimijoita. Törmäyksiä SHA-1-algoritmista ei ole vielä löytynyt [27].

Tiivistealgoritmin tuottama tulos salataan allekirjoittajan yksityisellä avaimella. Yksityinen avain on usein älykortilla, josta kerrotaan luvussa 3.4.1. Salattu tiiviste on allekirjoitus. Yleensä allekirjoitus liitetään alkuperäiseen dokumenttiin. Esimerkiksi XML-allekirjoituksissa tämä voidaan tehdä lisäämällä dokumentin perään <*Signature*>-elementti, jossa allekirjoitus sijaitsee. Allekirjoitukseen voidaan liittää myös metatietoa, kuten esimerkiksi varmenne.

Hyötyä allekirjoituksesta saadaan, kun allekirjoitetun dokumentin vastaanottaja voi allekirjoituksen avulla varmistua dokumentin aitoudesta ja allekirjoittaneen henkilön identiteetistä. Molempia ominaisuuksia vaaditaan kehittyneeseen sähköiseen allekirjoitukseen. Kuva 4 selittää allekirjoituksen vahvistuksen.



Kuva 4: Digitaalisen allekirjoituksen vahvistus.

Vahvistus koostuu kolmesta osasta:

1. Lasketaan alkuperäisestä tiedosta tiiviste. Tiivisteen laskeminen tapahtuu samalla tavalla kuin allekirjoitusta tehtäessä.
2. Puretaan allekirjoitettu tiiviste allekirjoittajan julkisella avaimella.
3. Vertaillaan kohtien 1. ja 2. tuloksia. Mikäli tulokset ovat samat, niin dokumenttia ei ole muutettu allekirjoituksen jälkeen.

Allekirjoittajan identiteetti tulisi tarkistaa. Jokainen yksityinen avain liittyy vain yhteen julkiseen avaimeen. Täten voidaan olettaa, että allekirjoitus on tehty yksityisellä avaimella, joka liittyy vahvistamisessa käytettyyn julkiseen avaimeen. Identiteetin varmistamiseksi on tarkistettava että:

1. varmenne ei ole sulkulistalla,
2. varmenteen allekirjoitusketju on kunnossa juurivarmentajaan asti, sekä
3. juurivarmentaja on luotettava.

Kohtien 1, 2 ja 3 toteutuessa voidaan turvallisesti olettaa että allekirjoittaja on kuka hän väittää olevansa.

3.2.3 Varmenteet

Varmenne on sähköinen dokumentti, mikä sisältää tietoa henkilöstä tai toimijasta, kuten palvelimesta tai organisaatiosta. Varmenne on julkista tietoa, joka tarkoittaa että varmenteet ovat usein julkisesti saatavissa.

Varmenteeseen liittyy julkinen avain ja tietoa sen omistajasta ja käyttötarkoituksesta. Mikäli käyttäjällä on hallussaan varmenteen lisäksi varmenteeseen liittyvä yksityinen avain, voi käyttäjä tunnistautua yksityisen avaimen avulla. Yksityinen avain ei ole osa varmennetta. Varmenne sisältää tietoa jota voi jakaa julkisesti. Yksityinen avain ei ole julkista tietoa. Yksityinen avain voi kuitenkin sijaita samassa tiedostossa kuin varmenne. Tässä tapauksessa tiedostoa ei tule jakaa julkisesti.

Yleisesti käytettävä varmenne on X509-varmenne. VRK:n myöntämät varmenteet ovat X509 V3 (versio 3) -varmenteita. X509 V3 -varmenne sisältää seuraavat tiedot [17]:

- Versio: käytettävän X509-varmenteen versio. Nykyään VRK:n myöntämät varmenteet ovat versiota 3.
- Varmentaja: CA (Certificate Authority) joka on myöntänyt varmenteen. Suomessa varmenteita terveydenhuollon henkilöille myöntää VRK.
- Sarjanumero: positiivinen kokonaisluku, joka yksilöi varmenteen CA:n järjestelmässä. Varmenteen yksilöivä tieto on CA:n nimi ja kortin sarjanumero.
- Käytetyt algoritmit: Algoritmit joilla CA on allekirjoittanut myönnetyn varmenteen.
- Allekirjoitus: CA:n luoma allekirjoitus, joka vahvistaa varmenteen aitouden.
- Validiteettiaika: Aikaväli jonka välissä varmennetta voi käyttää.
- Kohde (Subject): henkilö tai toimija, joka on yhdistetty varmenteen sisältämään julkiseen avaimeen.
- Kohteen julkisen avaimen tiedot: sisältää käytetyn algoritmin ja julkisen avaimen.
- Laajennoskentät (Extensions): kentät mahdollistavat ylimääräisten ominaisuuksien lisäämisen varmenteeseen. Laajennoskentät eivät ole pakollisia. Laajennoskentät voivat sisältää tietoa esimerkiksi varmenteen käyttötarkoituksesta tai ”Subject Alternative Name” -kentän, jota käytetään korttikirjautumisessa Windows-työasemille.

Varmenteiden yksityinen avain voi sijaita samassa tiedostossa kuin varmenne. Älykorttien tapauksessa yksityinen avain on tallessa kortin sirulla, eikä yksityistä avainta voi kortilta hakea.

3.3 Allekirjoitustyypit

Digitaalisista allekirjoituksista on useita standardeja, kuten PGP-allekirjoitukset tai XML-allekirjoitukset. Kaikki kuitenkin noudattavat samoja periaatteita ja askeleita. Yksi yleisimmistä allekirjoitustekniikoista on XML-allekirjoitus, jonka avulla voidaan allekirjoittaa XML-tiedostojen lisäksi binääritiedostoja. XML-allekirjoitus on XML-tiedosto, joka sisältää kaiken allekirjoituksen vahvistamiseen tarvittavan tiedon. Allekirjoitettu data voi kuitenkin olla omassa tiedostossaan, jolloin XML-allekirjoitus sisältää viitteen allekirjoitettavaan dataan. Luku 3.3.1 kertoo lisää XML-allekirjoituksesta.

XML-allekirjoituksien ominaisuuksia jatkaa Virossa luotu BDOC-formaatti. Tässä formaatissa on mahdollista pakata useita eri tiedostoja yhteen allekirjoitukseen ja luoda niille XML-allekirjoitus. BDOC-formaatista kerrotaan enemmän luvussa 3.3.2.

3.3.1 XML-allekirjoitus

World Wide Web Consortium (W3C) on luonut standardin XML-tiedostojen allekirjoittamiseen. Tässä luvussa tarkastellaan XML-allekirjoituksen allekirjoituselementtiä ja eri tapoja joilla allekirjoituselementin voi upottaa dokumentteihin. Luvun lopussa tarkastellaan datan standardisointia ennen allekirjoitusta.

Allekirjoituselementti

XML-allekirjoituselementti on määritelty lähteessä [7]. XML-allekirjoitus on *<Signature>*-elementin sisällä. Listauksessa 3.1 on yksinkertainen esimerkki allekirjoituksesta.

```
<Signature Id="MyFirstSignature" xmlns="http://www.w3.org/2000/09/xmldsig#">
   <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/2006/12/xml-c14n11"/>
      <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
      <Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126/">
         <Transforms>
           <Transform Algorithm="http://www.w3.org/2006/12/xml-c14n11"/>
         </Transforms>
         <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
         <DigestValue>dGhpcyBpcyBub3QgYSBzaWduYXR1cmUK.../DigestValue>
      </Reference>
   </SignedInfo>
   <SignatureValue>hQKtk4xBMFgONRAjMmWEhgLs33R4IJ0c0biHMrz93YU0Tz9bRn9//BMNz+VT4qGO-
HYBMUH2rSGk8/Ixg5sIlZ7oCyxt8sSHzQyUwWPPIjv95QTceyZo4RdtZ7gQI6orCYTuH9xIc-
mmhXs07jLflfgdA1WTcpMCsun16DHcIG20l5BjoQMduqdZ7bwWYBgEpTqF+RmKDf6ywYrYAYaVdu-
OslhDgLxKHnQsw3OgA7zPlUvlQRRRLyUlir1AkiemLqmjdZJ1Uyi7WOn/tNd1aL-
GCNVjJjjm8nQ4rSie9LI7JuURus2zu58LvEM0mbnxSKjjHzET0Phy3ZwXVQAcxLcUrXIAA==
   </SignatureValue>
   <KeyInfo>
     <KeyValue>
        <DSAKeyValue>
           <P>...</P><Q>...</Q><G>...</G><Y>...</Y>
        </DSAKeyValue>
      </KeyValue>
   </KeyInfo>
</Signature>
```

Listaus 3.1: XML-allekirjoituselementti. Itse allekirjoitus on <SignatureValue>-elementissä.

<*Signature*>-elementti sisältää kolme lapsielementtiä: <*SignedInfo*>, <*SignatureValue*> ja <*KeyInfo*>. Lista allekirjoitettavasta tiedosta on elementin <*SignedInfo*> sisällä. Kaikki <*SignedInfo*>-elementin sisällä on allekirjoitettavaa tietoa. Allekirjoitettava tieto sisältää viitteet dokumentteihin, allekirjoituksessa käytettävät algoritmit ja allekirjoitettavaa dokumenttia esikäsitelleet metodit.

Allekirjoituksen tuottama arvo on <*SignatureValue*>-elementissä ja **elementti ei ole allekirjoitettavaa tietoa**. Arvo saadaan, kun <*SignedInfo*>-elementin sisältämä tieto ajetaan tiivistys- ja allekirjoitusalgoritmien läpi. Elementin sisältö on käytännössä vain pitkä satunnaiselta vaikuttava merkkijono.

<*KeyInfo*>-elementissä on tieto avaimesta, minkä avulla allekirjoitettu tieto voidaan vahvistaa. Elementin sisältö voi olla esimerkiksi varmennetieto. Elementti ei ole pakollinen eikä sitä allekirjoiteta.

<*CanonicalizationMethod*>-elementti sisältää tavan, jolla <*SignedInfo*>-elementin sisältö normalisoidaan standardiin muotoon ennen tiivisteen laskemista. XML-tiedostojen normalisoinnista kerrotaan lisää myöhemmin tässä luvussa.

<*SignatureMethod*>-elementti sisältää tiedon algoritmista, jolla <*SignedInfo*> muutetaan <*SignatureValue*>-elementin arvoksi. <*SignatureMethod*>-elementti sisältää tiivistefunktion ja allekirjoitusalgoritmin.

<*Reference*>-elementin sisällä on viitteet allekirjoitettavaan dataan, datasta lasketun tiivisteen ja tiivistealgoritmi jolla tiiviste on laskettu. Näitä elementtejä voi olla useampi. Yhteen allekirjoitukseen voi sisällyttää esimerkiksi useamman HTML-sivun. Tämän lapsielementtinä voi olla <*Transforms*>, joka sisältää tiedon algoritmista joka tuotti syötteen tiivistealgoritmille. <*Transforms*>-elementin sisällä ovat askeleet jotka suoritetaan ennen tiivisteen laskemista. Tällaisia voivat olla tiedon standardointi standardiin formaattiin, koodaaminen tai purkaminen. Esimerkiksi XML-dokumentit voidaan normalisoida formaattiin jossa ei ole ylimääräisiä rivinvaihtoja tai kommentteja.

<*SignedInfo*>-elementin validointi on kaksiosainen:

1. Validoidaan jokainen <*Reference*>-elementti: tarkastetaan, että jokaisen referenssin tiiviste (joka on laskettu <*DigestMethod*>-metodilla) vastaa siitä laskettua tiivistettä, joka on <*DigestValue*>:n sisällä oleva arvo.
2. Validoidaan allekirjoitus: tarkistetaan, että <*SignatureValue*>-elementin sisältö vastaa tulosta, joka saadaan prosessoimalla <*SignedInfo*>-elementti käyttämällä <*CanonicalizationMethod*>- ja <*SignatureMethod*>-elementtien määräämällä tavalla.

XML-allekirjoituksen etu on sen joustavuus; sillä voi allekirjoittaa XML-tiedostojen lisäksi binääritiedostoja <*Reference*>-elementin avulla.

Enveloped, Enveloping ja detached -allekirjoitustyypit

XML-allekirjoituksia on kolmea eri tyyppiä: e*nveloped*, *enveloping* ja *detached*. E*nveloped*, *enveloping* ovat XML-tiedostoja ja XML-allekirjoitus sijaitsee samassa tiedostossa. *Detached*-allekirjoitus on erillisessä tiedostossa ja mahdollistaa muidenkin formaattien allekirjoittamisen.

Termi *enveloped* viittaa XML-allekirjoitukseen, jossa XML-allekirjoituselementti <*Signature*> on muun XML:n seassa. *Enveloped*-tyyppinen allekirjoituselementti on toisen elementin lapsielementtinä.

```
<!DOCTYPE Envelope [
  <!ENTITY ds "http://www.w3.org/2000/09/xmldsig#">
  <!ENTITY c14n "http://www.w3.org/TR/2001/REC-xml-c14n-20010315">
  <!ENTITY enveloped "http://www.w3.org/2000/09/xmldsig#enveloped-signature">
  <!ENTITY xslt "http://www.w3.org/TR/1999/REC-xslt-19991116">
  <!ENTITY digest "http://www.w3.org/2000/09/xmldsig#sha1">
]>
<Letter>
   <Return-address>ville.ylimannela@student.tut.fi</Return-address>
   <To>Me</To>
   <Message>Return to sender, address unknown, no such number</Message>
   <From>
      <ds:Signature xmlns:ds="&ds;">
         <ds:SignedInfo>
            <ds:CanonicalizationMethod Algorithm=
                "http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
            <ds:SignatureMethod Algorithm=
                "http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
            <ds:Reference URI="">
               <ds:Transforms>
                  <ds:Transform Algorithm="&enveloped;">
                  </ds:Transform>
               </ds:Transforms>
               <ds:DigestMethod Algorithm="&digest;"/>
               <ds:DigestValue></ds:DigestValue>
            </ds:Reference>
         </ds:SignedInfo>
         <SignatureValue>...</SignatureValue>
      </ds:Signature>
   </From>
   <Attach>sillykittens.bmp</Attach>
</Letter>
```

Listaus 3.2: Enveloped-tyyppinen allekirjoitus, jossa allekirjoituselementti on muun XML:n seassa.

Yllä olevassa listauksen 3.2 esimerkissä allekirjoitus on <*From*>-elementin lapsielementtinä. Allekirjoituselementti jätetään pois, kun XML:lle lasketaan tiivistettä ja allekirjoituksen arvoa, eli elementtien <*ds:DigestValue*> ja <*ds:SignatureValue*> sisältöjä. [4]

Termi *enveloping* viittaa allekirjoitukseen, jossa *<ds:SignatureValue>*-elementti sisältää allekirjoitettavan datan *<ds:Object>*-elementtiin käärittynä. Alla olevassa esimerkissä on allekirjoitettu teksti ”Hello, World!”. [5]

```
<?xml version="1.0" encoding="UTF-8"?>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod
         Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <ds:SignatureMethod
         Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    <ds:Reference URI="#obj">
      <ds:DigestMethod
          Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue/>
    </ds:Reference>
  </ds:SignedInfo>
  <SignatureValue>...</SignatureValue>
  <ds:Object Id="obj">Hello, World!</ds:Object>
</ds:Signature>
```

Listaus 3.3: Envoloping-tyyppinen allekirjoitus, jossa allekirjoitettava tieto on <Object>-elementin sisällä.

”Hello, World”-merkkijonon tilalla voisi olla pelkän merkkijonon sijaan XML-formaatissa olevaa tietoa.

Detached-tyyppisissä allekirjoituksissa allekirjoitus on omassa tiedostossaan tai sisarelementtinä samassa XML-tiedostossa. Alla on pelkkä allekirjoituselementti. Elementti on omassa tiedostossaan ja viittaaminen allekirjoitettuun dataan tulee *<ds:Reference>*-elementin sisään. Alla olevassa allekirjoituksessa on allekirjoitettu HTML-dokumentti osoitteessa ”http://www.w3.org/TR/xml-stylesheet” [6]

```
<?xml version="1.0" encoding="UTF-8"?>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod
         Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <ds:SignatureMethod
         Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    <ds:Reference URI="http://www.w3.org/TR/xml-stylesheet">
      <ds:DigestMethod
          Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue/>
    </ds:Reference>
  </ds:SignedInfo>
  <SignatureValue>...</SignatureValue>
</ds:Signature>
```

Listaus 3.4: Detached-tyyppinen allekirjoitus. Allekirjoitettava tieto löytyy URI:n osoittamasta paikasta.

Jos detached-tyypissä allekirjoitettava data on samassa tiedostossa, niin tiedosto on listauksen 3.5 mukainen.

```
<?xml version="1.0" encoding="UTF-8"?>
<internally-detached>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      ←Signaturekentät tähän→
  </ds:Signature>
  <document Id="data">
     <title> Hejsan Ulle!</title>
     <author>Ville Ylimannela</author>
     <date>04.12.2015</date>
     <content>
        <para>Hur mår du? Jag passerade min svenska kursen!</para>
     </content>
  </document>
</internally-detached>
```

Listaus 3.5: Detached-tyyppinen allekirjoitus. Allekirjoitettava tieto löytyy samasta tiedostosta.

Tässä tapauksessa allekirjoitus ja allekirjoitettava dokumentti ovat sisarelementteinä yhteisen <*internally-detached*>-juurielementin alla.

Normalisointi

XML:n normalisoinnin tarkoituksena on, että erilaisia XML-tiedostoja käsitellään samalla tavalla riippumatta kommenteista, turhista rivinvaihdoista ja XML-dokumentin sisältämistä XML-allekirjoituksista. Listaus 3.6 sisältää normalisoimattoman XML-dokumentin.

```
<?xml version="1.0"?>

<?xml-stylesheet   href="doc.xsl"
   type="text/xsl"   ?>
<!DOCTYPE doc SYSTEM "doc.dtd">

<doc>Hello, world!<!-- Comment 1 --></doc>

<?pi-without-data     ?>

<!-- Comment 2 -->

<!-- Comment 3 -->
```

Listaus 3.6: Esimerkki normalisointialgoritmille annettavasta syötteestä.

Listauksen 3.6 normalisoitu muoto on listauksen 3.7 mukainen.

```
<?xml-stylesheet href="doc.xsl"
   type="text/xsl"   ?>
<doc>Hello, world!</doc>
<?pi-without-data?>
```

Listaus 3.7: Listauksen 3.6 normalisoitu muoto ilman kommentteja.

Kommentit voidaan sisällyttää normalisoituun XML-dokumenttiin. Tässä tapauksessa listauksen 3.6 XML-dokumentti muuttuu listauksen 3.8 mukaiseksi.

```
<?xml-stylesheet href="doc.xsl"
   type="text/xsl"   ?>
<doc>Hello, world!<!-- Comment 1 --></doc>
<?pi-without-data?>
<!-- Comment 2 -->
<!-- Comment 3 -->
```

Listaus 3.8: Listauksen 3.6 normalisoitu muoto kommenttien kanssa.

Normalisointiin on tarjolla neljä eri tapaa: versiot 1.0 ja 1.1 sekä molemmista versiot jotka jättävät XML-kommentit pois. Yläpuolella olevat listaukset 3.7 ja 3.8 ovat normalisoitu versiolla 1.1. XML-allekirjoitusten toteutuksien tulee tukea jokaista neljää normalisointitapaa. [11]

3.3.2 BDOC

BDOC on viron Sertifitseerimiskeskuksen luoma digitaalisen allekirjoituksen standardi. Sertifitseerimiskeskus on tällä hetkellä Viron ainoa laatuvarmenteiden tarjoaja. Sertifitseerimiskeskus tarjoaa varmenteita Viron kansalaisten tunnistautumiseen ja digitaaliseen allekirjoittamiseen [8]. BDOC:n on tarkoitus korvata vanhempi Viron käyttöön luotu allekirjoitusstandardi DDOC. [9]

BDOC on ZIP-kansio, jossa on allekirjoitus, allekirjoitetut tiedostot ja edellisiin liittyvät metatiedot. BDOC-formaatti on ZIP-pakattu. Tästä syystä BDOC-tiedostot ovat pienempiä kuin edeltäjänsä DDOC-formaatin tiedostot. DDOC-formaattia olevien tiedostojen luomista tullaan tukemaan vain vuoteen 2016 asti. [9]

BDOC-formaatti

BDOC-formaatti on ZIP-paketti, jossa on mukana ennalta sovittuja tiedostoja ja tiedostorakenteita. Rakenne on aina seuraavan mukainen [10]:

```
Document1.pdf
Document2.doc
mimetype
META-INF/manifest.xml
META-INF/signatures1.xml
```

Listaus 3.9: BDOC-formaatin hakemistorakenne.

Mimetype-tiedostossa on aina sama sisältö pakkaamattomassa muodossa:

```
application/vnd.etsi.asic-e+zip
```

Hakemisto META-INF/ sisältää manifest.xml-tiedoston ja yhden tai useamman signatures.xml-tiedoston. Manifest.xml sisältää listan kaikista tiedostoista ja hakemistoista joita ZIP-paketissa on. Manifest.xml **ei** sisällä juuressa sijaitsevaa mimetype-tiedostoa tai META-INF-hakemiston sisältöä. [10]

Signatures1.xml:n juurielementtinä on aina *<asic:XAdESSignatures>*, jonka lapsielementteinä on luvussa 3.3.1 kuvattuja allekirjoituselementtejä. Yhdessä tiedostossa voi olla yksi tai useampi allekirjoitus. Myös allekirjoitustiedostoja voi olla yksi tai useampi. [10]

Kaikki BDOC-paketissa olevat tiedostot allekirjoitetaan, poislukien mimetype-tiedosto ja META-INF-hakemiston sisältö. Tästä huolimatta jokainen allekirjoitettava tiedosto täytyy eksplisiittisesti referoida *<Reference>*-elementin sisällä. Allekirjoitettava tiedosto tulee XML-allekirjoituksen *<Reference>*-elementin URI-attribuutin sisään. Yllä olevassa listauksessa 3.9 URI olisi ”Document1.pdf” tai ”Document2.doc”. [10]

3.4 Allekirjoituksen välineet

Digitaalinen allekirjoitus vaatii yksityisen avaimen. Turvallinen tapa säilyttää yksityistä avainta on älykortti. Älykortilla yksityinen avain sijaitsee turvallisesti sirulla ja sitä on helppo kuljettaa mukana. Älykortin käyttäminen vaatii siihen soveltuvan kortinlukijan ja kortinlukijaohjelmiston.

3.4.1 Älykortti

Allekirjoituksiin tarvitaan yksityinen avain, joka sijaitsee usein älykortilla. Joskus älykortista käytetään myös termejä sirukortti tai toimikortti. Jokaiseen korttiin liittyy oma PIN-numero (Personal Identification Number), minkä avulla pääsee käsiksi sirulla sijaitsevaan yksityiseen avaimeen. VRK:n myöntämät älykortit sisältävät yksityisen avaimen ja yksityiseen avaimeen liittyvän varmenteen.

Yksityinen avain sijaitsee älykortin sirulla ja sitä voi käyttää digitaalisiin allekirjoituksiin. Yksityistä avainta ei saa ulos kortilta, vaan avain on käytettävissä vain PIN-koodin avulla [20]. Yksityinen avain luodaan älykortin sisällä, eli älykortilla olevasta yksityisestä avaimesta ei ole kopiota kortin ulkopuolella [21]. Käyttäjä ei edes itse tiedä yksityistä avaintaan. Älykortti on kuin pieni tietokone, joka saa virran ja kellosignaalin kortinlukijasta kontaktipinnan yli. Allekirjoitettava tavujono lähetetään allekirjoitettavaksi kortille. Kortti tunnistaa käyttäjänsä PIN-koodilla, allekirjoittaa sille lähetetyn tavujonon salaamalla sen yksityisellä avaimella ja antaa salatun tavujonon takaisin kutsujalle. Kortti tekee kaikki yksityistä avainta käyttävät toiminnot sisäisesti, minkä avulla yksityinen avain pysyy turvassa kortilla.

Yksityistä avainta ei saa ulos kortilta sen rajapinnan avulla, mutta se on kuitenkin tehtävissä erilaisten hyökkäysten avulla. Hyökkäykset korttia vastaan voidaan jaotella kahteen kategoriaan: anturointiin ja passiivisiin ohjelmistohyökkäyksiin. [18]

Anturointihyökkäykset fyysisesti muokkaavat korttia, minkä avulla sen sisällä oleva tieto voidaan saada ulos. Anturointi vaatii kuitenkin jopa useita viikkoja laboratoriotyöskentelyä ja saattaa tuhota älykortin sen tuloksena. Anturoinnissa sirua suojaavat kerrokset poistetaan typpihapolla jonka jälkeen sirua voidaan tutkia erilaisilla laitteilla, kuten mikroskoopeilla. [12][18]

Passiiviset ohjelmistohyökkäykset ovat hyökkäyksiä esimerkiksi prosessoria, salausalgoritmeja tai protokollia vastaan. Nämä ovat monessa tapauksessa vaarallisempia, sillä ne ovat toistettavassa helposti jokaiselle samanlaiselle kortille [12]. Suomessa älykortteja kansalaisille myöntää väestörekisterikeskus. Väestörekisterikeskuksen myöntämät varmenteet ovat nähtävissä VRK:n nettisivuilla (http://vrk.fineid.fi/certsearchB.asp). Samasta paikasta löytyvät sekä yksityiskäyttöön myönnetyt varmenteet että terveydenhuollon varmennepalvelu.

3.4.2 Kortinlukija ja kortinlukijaohjelmisto

Älykortin lukeminen vaatii kortinlukijan ja kortinlukijaohjelmiston. Älykortit käyttävät ulkoista virtalähdettä ja kellosignaalia. Tästä johtuen älykortti vaatii aina kortinlukijan kommunikoidakseen ympäristön kanssa. Kun kortti työnnetään lukijaan painuvat kortinlukijan ja älykortin nastat toisiaan vasten. Virta ja kellosignaali kulkevat kortille kontaktipinnan yli. [18]

Useat ammattikäyttöön tarkoitetut kannettavat tietokoneet sisältävät kortinlukijan. Koneisiin joissa kortinlukijaa ei ole, on kortinlukijoita saatavilla USB-liitännällä. Kortinlukijan asentaminen ei tyypillisesti vaadi erillisen ajurin asentamista.

Lukijan lisäksi vaaditaan kortinlukijaohjelmisto. Kyseinen ohjelmisto toimii ajurina tietokoneen ja älykortin välillä. Kortinlukijaohjelmisto kysyy PIN-koodia, kun kortilla sijaitsevalla varmenteella allekirjoitetaan dataa. Kortinlukijaohjelmisto lähettää PIN-koodin ja allekirjoitettavan datan älykortille ja välittää allekirjoitetun datan takaisin kutsujalle. Kotimaisille VRK:n myöntämille älykorteille on saatavilla maksutta Fujitsu Oy:n mPollux Digisign Client [19].

3.5 WYSIWYS

WYSIWYS tulee englanninkielestä sanoista What You See Is What You Sign. Termillä viitataan siihen, että käyttäjän tulee tietää mitä hän allekirjoittaa. WYSIWYS on tärkeä ominaisuus digitaalisissa allekirjoituksissa. Tarkoituksena on estää käyttäjää allekirjoittamasta väärää tietoa. Väärän tiedon allekirjoittaminen voi johtaa oikeudellisiin ongelmiin.

Henkilön allekirjoittaessa paperista dokumenttia, hänen on helppo nähdä allekirjoittamansa tieto; se on fyysisenä kopiona. Digitaaliset allekirjoitukset kohdistuvat sarjaan bittejä – ykkösiä ja nollia tietokoneen kovalevyllä – joten allekirjoitettavan datan tulkitseminen on huomattavasti vaikeampaa. Ihminen ei voi fyysisesti nähdä allekirjoitettavia bittejä. Bittien esittäminen suoraan käyttäjälle on täysin hyödytöntä, sillä ihminen ei kykenisi niitä tulkitsemaan.

Kaikilla tietokoneelle tallennetuilla biteillä on oikea tapa tulkita niitä, eli muuttaa ne ohjelman avulla ihmisen ymmärtämään muotoon. Esimerkiksi pdf-tiedoston voi avata ascii-editorilla, mutta se olisi täynnä symboleita jotka eivät merkitsisi mitään lukijalle. Jos sama pdf-tiedosto avataan Adoben Acrobat Readerillä, se avautuu halutulla tavalla. Ongelmia on myös esimerkiksi doc-tiedostojen tulkitsemisessa, sillä tiedostot voivat näyttää hyvinkin erilaisilta riippuen ohjelmasta (Word, Open Office, Libre Office) ja ohjelman versiosta.

Tärkeä WYSIWYS-ominaisuuteen liittyvä termi on semanttinen välimatka (semantic distance). Semanttisella välimatkalla tarkoitetaan matkaa bittitason ja visualisaation välillä. Esimerkiksi MS Wordin doc-tiedostoilla on korkea semanttinen välimatka ja ASCII-tiedostoilla sekä bittikartoilla lyhyt. Semanttisen välimatkan kasvaminen tarkoittaa mahdollisten datan tulkinnasta johtuvien virhetilanteiden lisääntymistä. Semanttisen välimatkan pitäminen lyhyenä on tärkeää suunniteltaessa allekirjoituksia, sillä se vähentää datan tulkinnasta johtuvia ongelmia. Esimerkiksi doc-tiedosto voi näyttää täysin erilaiselta riippuen avataanko dokumentti Word 2000 vai LibreOffice -ohjelmistolla. Yksinkertaisen ascii-dokumentin voi avata miltei millä tahansa ascii-editorilla ja se näyttää aina hyvin samalta. Eroja voi kuitenkin edelleen esiintyä.

3.5.1 Ehdotus WYSIWYS-toteutuksesta

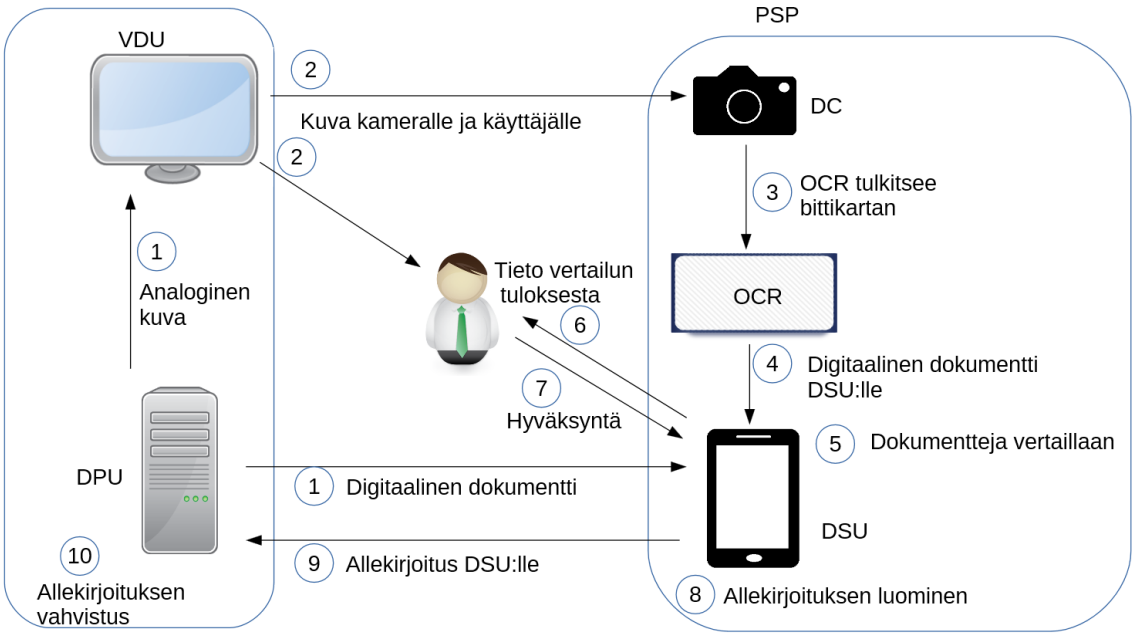
Audun Jøsang ja Bander AlFayyadh esittävät vuonna 2008 ratkaisuksi digital-to-analog-to-digital -muunnosta [14]. Tässä ehdotuksessa allekirjoittaja saa allekirjoitettavasta datasta kaksi saman tiedon sisältävää bittikarttaa. Toinen bittikartoista on digitaalisessa formaatissa ja toinen analogisessa. Analoginen bittikartta luetaan ja sen sisältöä vertaillaan digitaaliseen bittikarttaan. Mikäli sisältö on sama, voi käyttäjä olla varma että molemmat osapuolet tulkitsevat datan samalla tavalla. Tässä ratkaisussa analog-to-digital -muunnoksen tekevä kamera pitää huolen siitä, että allekirjoitat juuri sen tiedon minkä näet (WYSIWYS). [14]

Ehdotuksessa käytetään seuraavia komponentteja:

- DPU (Document Processing Unit) on dokumentin käsittelyssä käytettävä laitteisto ja ohjelmisto.
- VDU (Visual Display Unit) on monitori, joka näyttää käyttäjälle analogisena allekirjoitettavan bittikartan.

- DPP (Document Processing Platform) on kokonaisuus, joka sisältää VDU:n ja DPU:n.
- DC (Digital Camera) on kamera. Kameralla otetaan kuva VDU:n näyttämästä bittikartasta.
- OCR (Optical Character Recognition) on sovelluskomponentti, joka osaa tulkita analogisen kuvan ja muuttaa sen sisältämän tiedon digitaaliseksi.
- DSU (Digital Signature Unit) on laite jossa allekirjoituksen luominen tapahtuu.
- PSP (Portaple Secure Platform) on käyttäjään mukanaan kantava luotettava laite, joka sisältää DC:n, OCR:n ja DSU:n. PSP voi olla esimerkiksi älypuhelin tai tabletti, joka sisältää allekirjoitukseen tarvittavan korttipaikan tai muun turvallisen paikan yksityisen avaimen säilyttämiseen.

Kuva 5 esittää kyseisessä mallissa olevat askeleet.



Kuva 5: Kuvassa analogista kuvaa hyödyntävä WYSIWYS ratkaisu.

Prosessi on seuraava [14]:

1. DPU lähettää allekirjoitettavan tiedon digitaalisena DSU:lle ja analogisena kuvana VDU:lle.
2. Analoginen kuva on nyt näkyvillä käyttäjälle ja kameralle (DC).
3. DC ottaa kuvan VDU:n näytöltä ja luo kuvasta bittikartan. Bittikartta välitetään OCR:lle.
4. OCR muuttaa bittikartan digitaaliseksi dokumentiksi ja lähettää dokumentin eteenpäin DSU:lle.

5. DSU vertailee digitaalisia dokumentteja. Ensimmäinen digitaalinen dokumentti on saatu kohdassa 1 ja toinen kohdassa 4.
6. Jos digitaaliset dokumentit kohdassa 5 ovat samat, lähetetään käyttäjälle tästä tieto. Mikäli dokumentit eroavat toisistaan, lähetään käyttäjälle tieto että dokumentit eivät vastaa toisiaan. Allekirjoittaminen ei ole mahdollista jälkimmäisessä tapauksessa.
7. Jos dokumentit ovat kunnossa, voi käyttäjä käskeä DSU:n allekirjoittaa dokumentti.
8. DSU luo digitaalisen allekirjoituksen.
9. Digitaalinen allekirjoitus lähetetään DPU:lle.
10. DPU vahvistaa allekirjoituksen.

Kuvassa PSP voi olla esimerkiksi matkapuhelin tai muu mobiililaite. Matkapuhelimissa on nykyään valmiina kamera ja OCR esimerkiksi QR-koodien lukemiseen.

Esitetyssä ratkaisussa on kuitenkin useita käytännön haasteita. Merkittävin on se, että ratkaisu rajoittaa allekirjoitettavan datan formaattia ja kokoa. Ratkaisu kuitenkin soveltuu hyvin yksinkertaisen datan, kuten ostotransaktion, allekirjoittamiseen. Ratkaisusta ei ole vielä valmista toteutusta, joten toteutus tulisi tehdä alusta asti. Tämän lisäksi ratkaisussa tulee mahdollisesti vastaan käytettävyysongelmia, sillä ratkaisu lisää askelia jotka käyttäjän täytyy toteuttaa allekirjoituksen tehdäkseen.

3.5.2 WYSIWYS ja radiokemia

Jokainen merkkiaineen vapautus tapahtuu allekirjoituksella. Allekirjoituksella henkilö vahvistaa merkkiaineen turvallisuuden; sen voi injektoida ihmiseen. Tästä johtuen on tärkeää, että käyttäjä ymmärtää ja tietää mihin hän on laittamassa allekirjoitustaan. Jos WYSIWYS-ominaisuus ei toteudu, voi lopputuloksena olla tilanne, jossa käyttäjä:

1. laittaa allekirjoituksensa dataan josta hänellä ei ole täyttä tietoa: käyttäjä allekirjoittaa enemmän tai erilaista tietoa kuin mitä hänelle tietokoneen ruudulla näytetään [14]. Tämä voi johtaa tilanteeseen, jossa käyttökelvoton merkkiaine päätyy injektoitavaksi.
2. laittaa allekirjoituksensa dataan, joka on riittämätön. Tämä voi johtaa tilanteeseen, jossa käyttäjä ei ole vastuussa tekemästään virheestä, sillä hänen allekirjoittamansa data ei eksplisiittisesti kiellä merkkiaineen injektoimista. Tässä tapauksessa ongelma voi syntyä esimerkiksi oikeuden eteen mentäessä.

Allekirjoitettavan tiedon on oltava riittävä ja käyttäjän on oltava tieto siitä mitä hän on allekirjoittamassa.

4. ALKUPERÄINEN TOTEUTUS

PETERP-projektissa on tehty toteutus digitaalisista allekirjoituksissa. Tässä luvussa esitellään alkuperäinen projektissa tehty toteutus. Alkuperäinen toteutus on tehty ennen diplomityön kirjoittamista. Luvussa 5 tarkastellaan tarkemmin luvussa 4 esitellyn alkuperäisen toteutuksen ongelmia.

Käytössä on Matrix Gemini LIMS josta kerrotaan enemmän luvussa 4.1.1. Matrix Gemini ei tue digitaalisia allekirjoituksia. Tästä syystä digitaaliset allekirjoitukset on jouduttu toteuttamaan projektissa. Matrix Gemini LIMS mahdollistaa lisäosien ajon, jotka ovat .NET 3.5 kirjastoja.

4.1 Teknologiapakka

Toteutuksessa käytetään Microsoftin teknologioita, jotka kuvataan taulukossa 3.

Taulukko 3: Käytetyt teknologiat.

IIS	Internet Information Services on Microsoftin palvelinohjelmisto, jonka avulla voi tarjota asiakasohjelmistoille WCF-palveluita tai nettisivuja.
WCF	Windows Communication Foundation on WWW-sovelluspalvelu (Web service) -teknologia, jonka avulla asiakassovellukset voivat keskustella palvelimen kanssa.
WPF, Windows Forms	Windows Presentation Foundation ja Windows Forms ovat käyttöliittymäsuunnittelussa käytettäviä teknologioita.
Entity Framework	Entity Framework on .NET ympäristössä käytettävä kirjasto, jonka avulla tietokannan rivit voidaan liittää suoraan osaksi ohjelmatason oliota.
.NET 4.5, 3.5	.NET on Microsoftin ohjelmistokomponenttikirjasto. Palvelimella on käytössä .NET ympäristön versio 4.5. Ympäristössä kehitetyt ohjelmat suoritetaan Common Language Runtime -ympäristössä (CLR).

Teknologiavalinnat on tehty kehittäjillä olevan kokemuksen ja soveltuvuuden perusteella. Asiakkaalla on käytössään Windows-ympäristö, joten Microsoftin teknologiat ovat luonnollinen valinta.

4.1.1 Matrix Gemini LIMS

Projektissa käytettävä Matrix Gemini LIMS on laboratorion toiminnan ohjaukseen tarkoitettu järjestelmä. Projektissa tehtävä Matrix Gemini -konfiguraatio ei ole vielä tuotantokäytössä.

Matrix Gemini on helposti muokattava. Pääkäyttäjä voi muokata, lisätä ja poistaa ikkunoita, nappeja ja muita UI-elementtejä. Konfigurointi tapahtuu hyvin samalla tavalla kuin Windows Forms -ohjelman luonti, jossa käyttäjä voi vetää haluamansa UI-komponentit käyttöliittymään. Komponenteille on valmiiksi annettuja toimintoja, jotka voivat olla esimerkiksi:

- tietokannan kenttien arvojen näyttäminen ja editointi,
- muutoksien tallennus,
- uuden ikkunan avaaminen,
- näytteiden vastaanotto,
- laboratorioprotokollien luominen, sekä
- lisäosan laukaisu.

Matrix Gemini sisältää valmiita toimintoja, jotka liittyvät laboratorion tarvitseman toiminnallisuuden rakentamiseen. Matrix Geminin perustoiminnallisuutta ovat mm.: valmistusprotokollan ja testien määritteleminen, pöytäkirjaus avaaminen, tulosten kirjaaminen ja raporttien luominen.

Lisäosat ovat .NET 3.5 -kirjastoja. Lisäosat voivat piirtää käyttöliittymän, hakea tietoa tietokannasta ja tehdä palvelinkutsuja verkon yli. Lisäosille voi antaa käyttöliittymästä parametreja, kuten merkkiaine-erän eräkoodin. Matrix Geminissä ei ole Suomen lain täyttävää digitaalista allekirjoitusmahdollisuutta, joten allekirjoitus on toteutettu lisäosana.

4.1.2 PETERP

PETERP-järjestelmä on PET-kuvantamiseen toteutettu toiminnanohjausjärjestelmä. PETERP hallitsee kuvan 1 toiminnallisuuden pois lukien radiokemian laboratorion toiminnot. PETERP on integroitu toimimaan yhdessä Matrix Geminin kanssa. PETERP-järjestelmä on otettu tuotantokäyttöön vuoden 2015 alussa.

PETERP vastaanottaa lähetteitä sairaalan järjestelmistä. Jokainen lähete käsitellään lääkärin toimesta, joka valitsee lähetteeseen sopivan kuvantamisprotokollan. PETERP-ohjelmistossa on projektinhallinta, jonka avulla PET-keskuksen tutkijat voivat luoda tieteellisiä tutkimusprojekteja ja tilata eläin- tai ihmiskuvauksia.

Käsitellyt lähetteet ja tilatut tieteelliset kuvaustutkimukset aikataulutetaan ja vahvistetaan kuvattaviksi. Jokaisesta aikataulutetusta kuvauksesta muodostuu merkkiainevarauksia

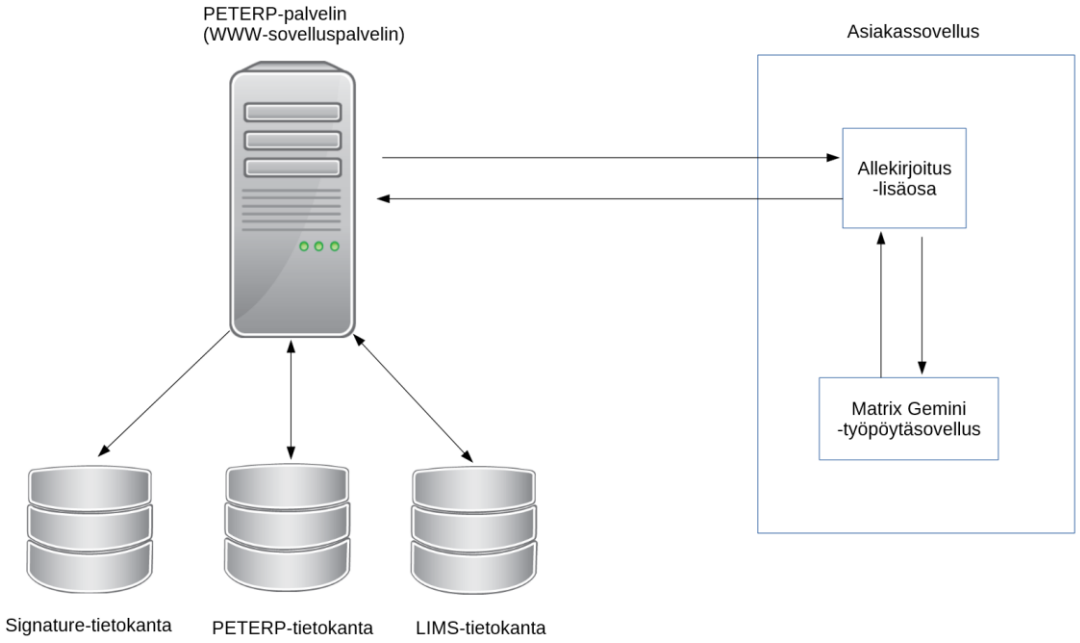
kalenteriin. Kalenterivaraukset auttavat radiokemian laboratorion henkilöstöä suunnittelemaan merkkiaineiden valmistusta etukäteen. Merkkiainevaraukset päätyvät RK-laboratorion (radiokemian laboratorion) tilauslistalle.

PETERP järjestää pyydetyt tutkimukset päivän ja pyydetyn laitteen mukaan, luoden jokaiselle päivälle oman kuvauskalenterin. Jokaisen kuvauksen aikana täytetään kuvauslomake, johon kirjataan kuvaukseen liittyvät tiedot. Kuvauksen lopuksi lomakkeelle täytetyt tiedot allekirjoitetaan digitaalisesti.

4.1.3 Ohjelmistoarkkitehtuuri

Ohjelmiston arkkitehtuuri on yksinkertainen asiakas-palvelinmalli. Kuvassa 6 on esitetty työn kannalta kriittisimmän komponentit. Kokonaisuus koostuu seuraavista komponenteista:

- **Asiakassovellus** koostuu Matrix Gemini -työpöytäsovelluksesta ja siihen tehdyistä lisäosista. Lisäosat ovat dll-tiedostoja, joita voi laukaista Matrix Gemini -sovelluksen käyttöliittymästä nappia painamalla.
- **PETERP-palvelin (WWW-sovelluspalvelin)** tarkistaa käyttäjän oikeuden tehdä allekirjoitus, tarkistaa allekirjoituksen ja tallentaa allekirjoituksen tietokantaan.
- **Signature-tietokanta** sisältää digitaaliset allekirjoitukset. Tietokannan skeema on luvussa 4.2.1 esitettävän kuvan 9 mukainen.
- **PETERP-tietokanta** sisältää hyväksytyt kortit ja niihin liittyvät allekirjoitusoikeudet. PETERP-tietokannasta tarkistetaan käyttäjän oikeus allekirjoittaa allekirjoitettu data. Tietokanta sisältää myös kaiken PETERP-järjestelmän tiedon, mutta tätä tietoa ei käytetä RK:n allekirjoituksien yhteydessä.
- **LIMS-tietokanta** sisältää Matrix Gemini -sovelluksen datan. LIMS-tietokannasta palvelin voi etsiä tietoa esimerkiksi merkkiaine-erän tilasta.



Kuva 6: Ohjelmistoarkkitehtuurin kuvaus.

Matrix Geminin lisäosatoiminnallisuus mahdollistaa toiminnallisuuden laajentamisen itse kirjoitetulla .NET 3.5 ohjelmakoodilla. Palvelin käyttää NET versiota 4.5, jonka avulla saadaan käyttöön kaikki uusimmat kirjastot ja ominaisuudet. Kommunikointi palvelimen ja asiakkaan (LIMS lisäosan) välillä tapahtuu verkon yli, joten eri NET versiot eivät aiheuta ongelmia.

Digitaalisen allekirjoituksen tapahtumasekvenssi on seuraava:

1. Käyttäjä painaa ”Sign”-nappia käyttöliittymästä. Allekirjoituskäyttöliittymät ovat nähtävissä kuvissa 10, 11 ja 12.
2. Nappi laukaisee allekirjoituslisäosan ja luo allekirjoituksen saamansa parametrien avulla.
3. Lisäosa lähettää tiedot verkon yli palvelimelle.
4. Palvelin tarkistaa allekirjoittajan oikeudet tehdä kyseinen allekirjoitus.
5. Palvelin tarkistaa, että allekirjoitus vastaa lähetettyä dataa.
6. Jos kohdat 4 ja 5 ovat kunnossa, tallennetaan allekirjoitus Signature-tietokantaan. Jos ei, annetaan kutsujalle poikkeus.

Palvelin palauttaa asiakkaalle tiedon toiminnon onnistumisesta normaalilla paluuarvolla tai kertoo epäonnistumisesta heittämällä poikkeuksen. Poikkeukset sisältävät viestin, joka kuvaa tapahtunutta ongelmatilannetta.

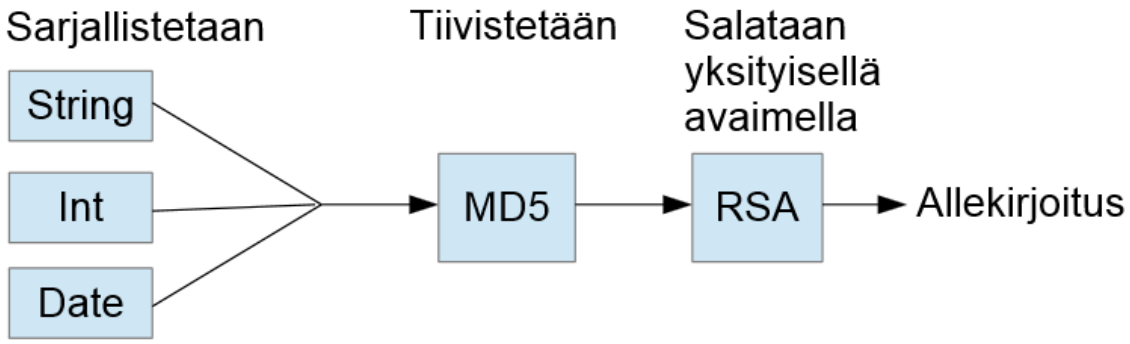
4.1.4 Allekirjoituksen toteutus

Digitaaliset allekirjoitukset tapahtuvat luvussa 3.2.2 kuvatulla tavalla. Järjestelmässä voi käyttää mitä tahansa VRK:n myöntämää älykorttia, joka on lisätty järjestelmään. Tämä sisältää myös kansalaisvarmenteet joita myönnetään Suomen kansalaisille. Vain järjestelmään eksplisiittisesti lisätyillä korteilla voi tehdä allekirjoituksia, eli pelkkä luotettu juurivarmentaja ei riitä.

Korttien lisääminen järjestelmään tapahtuu PETERP-ohjelmiston kautta. Kortin lisäämisen yhteydessä käyttäjän kortille määritellään oikeudet. Oikeudet ovat valittavissa allekirjoituspiste- ja merkkiainekohtaisesti, kuten luvussa on 2.2.1 esitelty.

Asiakassovelluksen toteutus

Allekirjoitus tehdään asiakaspäässä kuvan 7 prosessin mukaisesti.



Kuva 7: Asiakkaan allekirjoitusprosessi.

Prosessi jaetaan kolmeen osaan:

1. Sarjallistaminen: allekirjoitettava tieto sarjallistetaan allekirjoitettavaan muotoon, eli tavujonoksi. Sarjallistamiseen on projektissa tehty toteutus, joka sarjallistaa listan erityyppisiä C#-ohjelmointikielen olioita. Oliot voivat olla mm. merkkijonoja, kokonaislukuja ja päivämääriä.
2. Tiivistäminen: tieto ajetaan tiivistefunktion läpi. Käytössä on MD5-algoritmi.
3. Salaus: allekirjoitetaan tiiviste (tavujono) yksityisellä avaimella käyttäen RSA-algoritmia. Salauksen yhteydessä tieto tiivistetään uudelleen SHA-1 -algoritmilla.

Lopuksi allekirjoitus lähetetään palvelimelle käyttäen WWW-sovelluspalvelua. Sarjallistaminen ja tiivistäminen esitellään luvussa 4.1.5. Salaus ja salauksen yhteydessä tehtävä tiivistäminen esitellään luvussa 4.1.6.

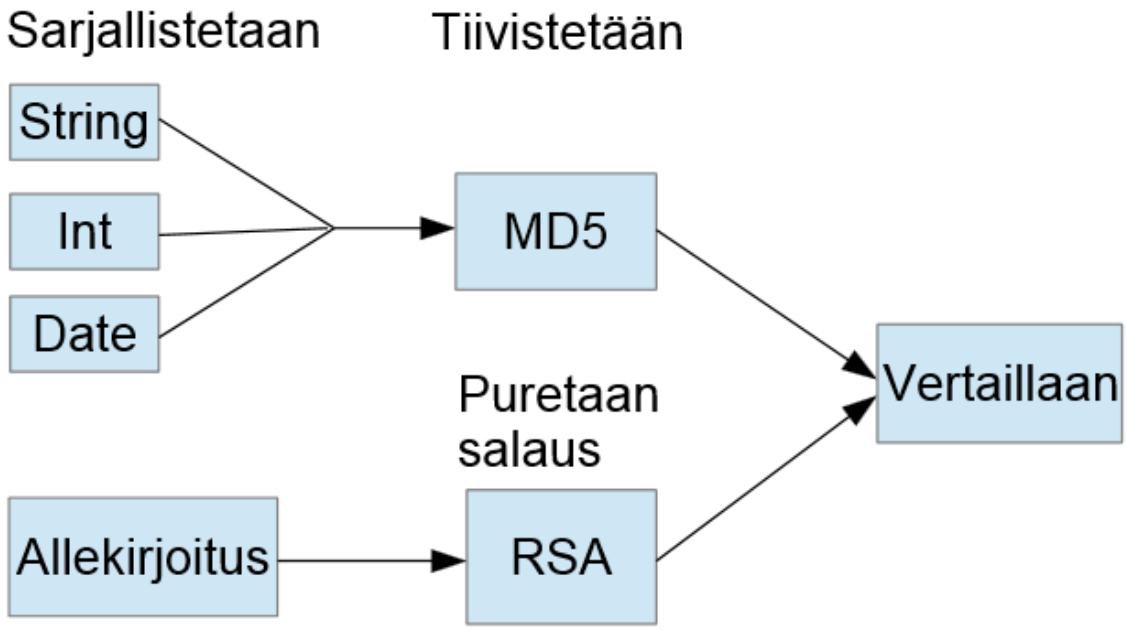
Palvelintoteutus

Palvelintoteutus on WWW-sovelluspalvekutsu, joka saa parametrinaan kuvassa 9 esitetyn DigitalSignature-kantaluokasta perityn olion. Toteutus koostuu seuraavista askeleista:

1. Allekirjoitusoikeuksien tarkistus: tarkistetaan että allekirjoituksen luonut käyttäjä saa allekirjoittaa allekirjoitetun tiedon. Allekirjoitusoikeudet ovat merkkiaine- ja allekirjoituspistekohtaisia.
2. Allekirjoitusketjun tarkistaminen: allekirjoitukset tulee antaa oikeassa järjestyksessä. Merkkiaine-erien allekirjoitukset annetaan järjestyksessä synteesi, käyttöön vapautus, QA-vapautus ja QA-hyväksyntä.
3. Allekirjoituksen vahvistus: vahvistetaan että allekirjoitettua tietoa tai allekirjoitusta ei ole muokattu.
4. Tallennetaan allekirjoitus: onnistunut allekirjoitus tallennetaan Signature-tietokantaan.

Jos jokin kohdista 1-3 epäonnistuu, allekirjoitusprosessi keskeytyy ja asiakasohjelmistolle annetaan virhettä kuvaava virheilmoitus. Tarkistukset ovat tarpeellisia, sillä palvelintoteutuksen tulee operoida oletuksella, että jokainen kutsu saattaa olla hyökkäys tai käyttäjän tekemä virhepainallus. Kaikkien tarkastusten ollessa kunnossa data tallennetaan tietokantaan.

Kohdan 3 allekirjoituksen vahvistus tapahtuu alla olevan kuvan 8 mukaisesti. Tässä tapauksessa lähtötietoja on enemmän kuin allekirjoituksen teossa, sillä alkuperäisen datan lisäksi on allekirjoitus, jonka aitous halutaan vahvistaa.



Kuva 8: Allekirjoituksen vahvistaminen palvelimella.

Vahvistus tapahtuu seuraavasti:

1. Sarjallistaminen: sarjallistaminen tehdään projektissa toteutetulla funktiolla, joka sarjallistaa C#-olioita tavujonoksi.

2. Tiivistäminen: lasketaan sarjallistamisen tuloksena saadun tavujonon tiiviste käyttämällä MD5-tiivistealgoritmia.
3. Allekirjoituksen purkaminen: puretaan allekirjoitus käyttämällä RSA-algoritmia ja allekirjoittajan julkista avainta.
4. Arvojen vertailu: verrataan laskettua tiivistettä (kohta 2) purettuun allekirjoitukseen (kohta 3). Jos nämä kaksi vastaavat toisiaan, allekirjoitus pitää paikkansa. Tieto on allekirjoitettu käyttäjän yksityisellä avaimella ja allekirjoitettavaa tietoa ei ole muutettu.

Sarjallistamisessa on hyvä huomioida, että se tulee tehdä samalla tavalla sekä palvelimella että asiakasohjelmistossa. Jos sarjallistettavaa tieto on kolmessa eri oliossa, tulee oliot syöttää sarjallistajafunktiolle samassa järjestyksessä palvelimella ja asiakasohjelmistossa.

4.1.5 Allekirjoitettavan tiedon sarjallistamisfunktio

Nykyinen toteutus mahdollistaa C#-olioiden sarjallistamisen tavujonoksi. Sarjallistaminen tapahtuu käyttämällä .NET-ympäristön MemoryStream ja BinaryFormatter -luokkia. Sarjallistaminen tehdään listauksen 4.1 funktiolla.

```
public static byte[] CalculateHash(params object[] data)
{
   using (MemoryStream ms = new MemoryStream())
   {
      BinaryFormatter bf = new BinaryFormatter();
      foreach (var item in data)
      {
         bf.Serialize(ms, item);
      }

      MD5 md5 = MD5.Create();
      return md5.ComputeHash(ms.ToArray());
   }
}
```

Listaus 4.1: Sarjallistamisessa käytetty ohjelmakoodi.

Funktio saa parametrina n-kappaletta C#-kielen olioita. Oliot muutetaan tavujonoksi, jonka jälkeen tavujonosta lasketaan MD5-tiiviste. Tiivistäminen on tässä turha vaihe, sillä tiivistäminen tehdään myöhemmin uudestaan allekirjoituksen yhteydessä. Tällä hetkellä allekirjoitettava data tiivistetään kahdesti: kerran MD5-algoritmilla ja allekirjoituksen yhteydessä SHA1-algoritmilla (listaus 4.2). Ensimmäinen tiivistys on redundantti ja sen voi tarvittaessa jättää korjatun toteutuksen ulkopuolelle. Tiivistämisestä kahteen kertaan on haittaa, sillä MD5-tiiviste merkittävästi heikentää allekirjoituksen vahvuutta. Sarjallistamisfunktion ongelmista kerrotaan luvuissa 5.2 ja 5.2.1.

4.1.6 Allekirjoituksen tiivistäminen, luominen ja vahvistaminen

Allekirjoitettavan tiedon tiivistäminen, allekirjoituksen luominen ja allekirjoituksen vahvistaminen tapahtuvat käyttämällä .NET 3.5 -komponenttikirjaston tarjoamia valmiita luokkia.

Allekirjoituksen luominen on esitetty listauksessa 4.2. Allekirjoituksen luominen alkaa hakemalla allekirjoitusvarmenne Windowsin varmennevarastosta. Allekirjoituksessa käytetään RSA-algoritmia. Allekirjoituksen luominen tapahtuu komponenttikirjaston tarjoaman *RSACryptoServiceProvider*-luokan *SignData*-jäsenfunktiossa. *SignData*-jäsenfunktio ottaa parametrikseen allekirjoitettavan tavujonon ja allekirjoitettavan datan tiivistämiseen käytettävän algoritmin. *SignData*-jäsenfunktio tiivistää ja allekirjoittaa saamansa tavujonon. Allekirjoituksen yhteydessä funktio kysyy käyttäjältä kortin allekirjoitusavainta suojaavan PIN-koodin.

```
public static DigitalSignatureData CreateDigitalSignature(byte[] data)
{
   X509Certificate2 cert = CertificateReader.GetDigitalSignatureCertificate();

   RSACryptoServiceProvider rsa = null;
   try
   {
      rsa = (RSACryptoServiceProvider)cert.PrivateKey;

      byte[] signature = rsa.SignData(data, new SHA1CryptoServiceProvider());
      byte[] cspBlob = rsa.ExportCspBlob(false);
      string name = cert.Subject;

      return new DigitalSignatureData(signature, cspBlob, name);
   }
   finally
   {
      if (rsa != null)
      {
         rsa.Clear();
      }
   }
}
```

Listaus 4.2: Allekirjoituksen luominen.

CreateDigitalSignature-funktio palauttaa käyttäjälle allekirjoituksen, julkisen avaimen ja allekirjoittajan nimen käyttäen apuluokkaa *DigitalSignatureData*.

Allekirjoituksen vahvistaminen tapahtuu listauksen 4.3 *VerifyData*-funktiolla. Funktio saa parametrikseen sarjallistetun tiedon, allekirjoituksen ja julkisen avaimen. Koodissa julkisesta avaimesta käytetään nimeä *cspBlob*.

```
public static bool VerifyData(byte[] data, byte[] signature, byte[] cspBlob)
{
      RSACryptoServiceProvider rsa = new RSACryptoServiceProvider();
      rsa.ImportCspBlob(cspBlob);

      return rsa.VerifyData(data, new SHA1CryptoServiceProvider(), signature);
}
```

Listaus 4.3: Allekirjoituksen vahvistaminen.

Vahvistus tapahtuu .NET-komponenttikirjaston tarjoaman *RSACryptoServiceProvider*-luokan *VerifyData*-jäsenfunktion avulla. Jäsenfunktio saa parametreikseen allekirjoitettavan datan, tiivistefunktion ja vahvistettavan allekirjoituksen. Jäsenfunktio palauttaa boolean-arvon *true* jos allekirjoituksen vahvistaminen onnistuu.

4.2 Allekirjoitettava tieto

Teknisellä tasolla allekirjoitettava tieto on jono bittejä tietokoneen muistissa tai kovalevyllä, jotka sarjallistetaan binäärimuotoon, tiivistetään ja allekirjoitetaan. Toiminnallisella tasolla allekirjoitettava tieto on jaettavissa loogisiin kokonaisuuksiin. LIMS-järjestelmässä digitaalisia allekirjoituksia tarvitaan kolmeen kokonaisuuteen:

1. Protokollien hyväksyntä: allekirjoitettava data on XML-merkkijono. XML:n sisältönä ovat protokollan perustiedot ja lista sen sisältämistä testeistä.
2. Merkkiainetilauksen hyväksyntä tuotantoon: allekirjoitettava data on XML-merkkijono. XML sisältää listan merkkiainetilauksia.
3. Merkkiainetuotantoprosessin allekirjoitukset: tässä allekirjoitetaan taulukossa 5 esiteltyä tietoa. Tällä allekirjoituksella käyttäjä vakuuttaa, että vaihe on tehty kunnolla ja voidaan edetä prosessissa eteenpäin. Tässä ei allekirjoiteta tuloksia, vaan allekirjoituksella käyttäjä vakuuttaa henkilökohtaisesti, että vaihe on tehty oikein ja ohjeiden mukaisesti.

Kohta 1 liittyy merkkiaineprotokolliin. Kohdat 2 ja 3 liittyvät merkkiaine-eriin. Kohta 2 eroaa kohdista 1 ja 3 siten, että yksi allekirjoitus voi sisältää useamman merkkiaine-erän.

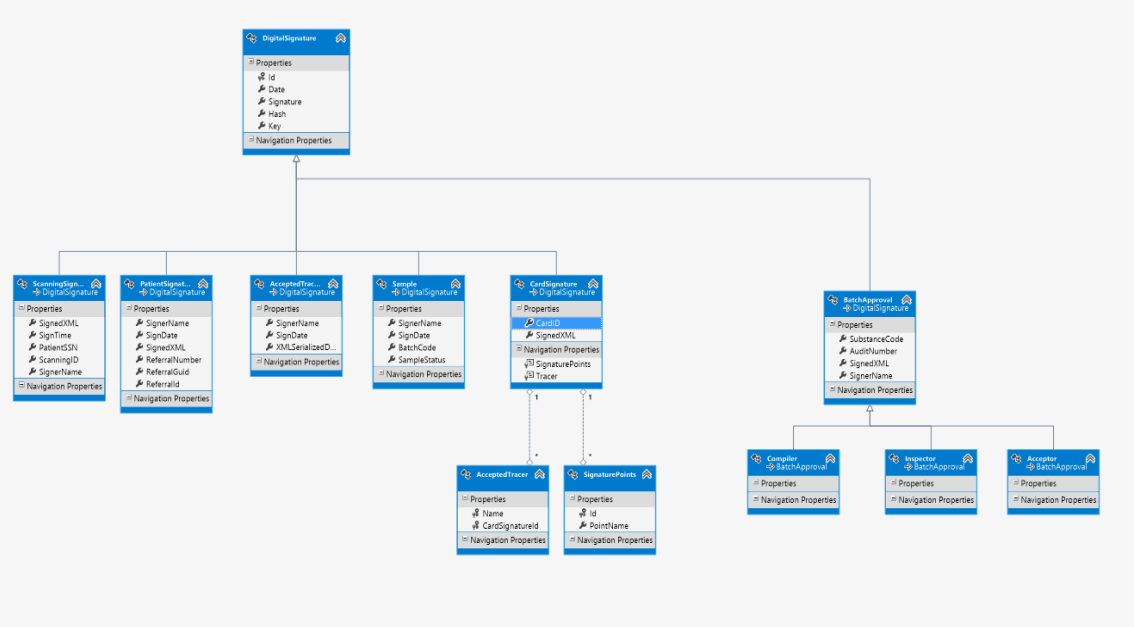
4.2.1 Tietomalli

Tietomalli kuvaa millaista tietoa tietokantaan tallentuu. Digitaalisten allekirjoitusten tietokantaan tallentuva tieto on jaettu useaan eri luokkaan. Kuvassa 9 esitellään projektissa käytössä oleva digitaalisten allekirjoitusten tietomalli.

Kuvassa 9 näkyvien luokkien jäsenmuuttujat tallennetaan suoraan tietokantaan. Tietokantaan tallentaminen tapahtuu Entity Framework -kirjaston avulla. Kirjasto linkittää C#-luokat automaattisesti SQL-tietokantatauluihin. Kirjasto osaa hakea, muuttaa, tallentaa ja poistaa ohjelmakoodissa muokattuja C#-luokkia SQL-tietokannasta. Kirjasto vähentää

tarvetta SQL-lauseiden kirjoittamiselle. Kuvan C#-oliot eivät sisällä vierasavainviitteitä, vaikka ne ovat olemassa tietokannassa.

Jokainen allekirjoitus periytyy yhteisestä kantaluokasta *DigitalSignature*. Merkkiainetuotantoprosessin allekirjoitukset tallennetaan *Sample*-tietokantatauluun, merkkiaineen tuotantoon hyväksymisallekirjoitukset tallennetaan *AcceptedTracerOrder*-tauluun ja protokollien hyväksymisallekirjoitukset tallennetaan *BatchApproval*-taulusta periytettyihin tauluihin.



Kuva 9: Allekirjoitusten tietomalli. Kaikki allekirjoitukset periytyvät kantaluokasta, jossa on kaikille yhteisiä pakollisia kenttiä.

Taulut *ScanningSignature*, *PatientSignature* ja *CardSignature* liittyvät PETERP-järjestelmään ja täten jäävät tämän työn ulkopuolelle. Muutoksia tehtäessä taulut on kuitenkin huomioitava, sillä LIMS-allekirjoituksiin tehtävät muutokset eivät saa rikkoa PETERP-järjestelmän allekirjoituksia. Jos esimerkiksi kantaluokkaan tulee muutoksia, tulee muutos vaikuttamaan jokaiseen allekirjoitukseen.

Käytetty rakenne mahdollistaa erityyppisten allekirjoitusten haun tietokannasta. Hakuja voidaan tehdä esimerkiksi *Sample*-tauluun siinä olevien jäsenmuuttujien perusteella. On mahdollista etsiä *Sample*-tyyppistä allekirjoitusta eräkoodin ja allekirjoitusajan perusteella.

Jokainen allekirjoitus periytyy kantaluokasta *DigitalSignature*. Kantaluokka sisältää jokaiselle allekirjoitukselle yhteistä perustietoa. Kantaluokan sisältämä tieto on kuvattu taulukossa 4.

***Taulukko 4: DigitalSignature*-luokka on allekirjoitusten kantaluokka.**

Attribuutti	Tyyppi	Kuvaus
Id	Integer32	Taulun pääavain, joka on automaattisesti kasvava 32-bittinen kokonaisluku.
Date	DateTime	Date-kenttään tallentuu allekirjoituksen tallennus päivämäärä.
Signature	ByteArray	Signature-kenttä on allekirjoituksen arvo. Vastaa XML-allekirjoituksen <*SignatureValue*>-elementtiä.
Hash	ByteArray	Hash-kenttä on sarjallistetusta tiedoista laskettu tiiviste. Tiiviste lasketaan sarjallistamalla allekirjoitettava tieto ja tiivistämällä se MD5-tiivistefunktiolla.
Key	ByteArray	Key-kenttä on julkinen avain, jolla allekirjoitus on luotu. Kenttä voidaan linkittää PETERP-tietokannassa olevaan julkiseen avaimeen, jossa on listattu hyväksytyt allekirjoitusvarmenteet ja niiden tekemät operaatiot.

DigitalSignature-kantaluokasta periytetyt luokat sisältävät kahdenlaista tietoa.

1. Allekirjoitettava tieto: data joka sarjallistetaan binäärimuotoon ja annetaan syötteeksi MD5-funktiolle. Allekirjoitettava data voi olla useammassa tietokannan sarakkeessa.
2. Metatieto: tietoa, jonka avulla allekirjoituksia voi hakea tietokannasta.

Sarake voi olla sekä allekirjoitettavaa dataa että metatietoa. Näin on tehty merkkiainetuotantoprosessin allekirjoitukset kohdalla: BatchCode-jäsenmuuttuja sarjallistetaan tavujonoksi ja allekirjoitetaan, mutta samaa jäsenmuuttujaa käytetään myös tietokantahakuihin. Muissa kuvan 9 tapauksissa allekirjoitettava tieto on sarjallistettu XML-merkkijonoksi ennen allekirjoitusta, joka johtuu siitä, että allekirjoitettava tieto on muissa allekirjoituksissa rakenteellista. BatchApproval- ja AcceptedTracerOrder-allekirjoituksissa allekirjoitetaan XML-sarjallistettua tietoa.

Merkkiainetuotantoprosessin allekirjoitukset: Sample-taulu

Merkkiainetuotantoprosessin aikana allekirjoitetaan synteesi, merkkiaineen vapautus ja laatukontrollit. Tuotantoprosessin allekirjoitus on yksinkertainen vahvistus että prosessi saa edetä eteenpäin. Jos synteesi epäonnistuu tai laadunvahvistuksessa havaitaan ongelmia, ei näitä allekirjoituksia anneta. Allekirjoitettavaa tietoa ovat allekirjoittajan nimi,

allekirjoituspäivämäärä, eräkoodi sekä allekirjoitettava tuotantoprosessin vaihe. Taulukossa 5 on kuvattu Sample-allekirjoituksien tietosisältö.

Taulukko 5: Sample-luokka on merkkiainetuotantoprosessin allekirjoituksissa käytetty luokka.

Attribuutti	Tyyppi	Kuvaus	Allekirjoitetaan
SignerName	NVARCHAR(MAX)	Allekirjoittajan nimi	Kyllä
SignDate	DateTime	Allekirjoitusaika	Kyllä
BatchCode	NVARCHAR(MAX)	Allekirjoitettavan erän eräkoodi	Kyllä
SampleStatus	NVARCHAR(MAX)	Kuitattu tieto: { SynthesisOperator | Release | QaRelease | QAApproval }	Kyllä

Ei-XML -pohjaiseen toteutukseen on päädytty, koska Sample-allekirjoitukset ovat yksinkertaisia vahvistuksia. Sample-allekirjoitukset eivät sisällä monimutkaista rakenteellista tietoa. Sample-tyyppisiä allekirjoituksia käytetään merkkiaine-erien tuotantoprosessissa synteesin ja laadunvalvonnan kohdalla. Sample-allekirjoitukset kertovat onko vaihe suoritettu onnistuneesti vai ei. Jokainen allekirjoitettava alkio on oma tietokantasarakkeensa, joten niiden avulla voi tehdä suoraan hakuja.

Merkkiainetilausten hyväksyntä: AcceptedTracerOrder-taulu

Merkkiainetilausten hyväksynnässä allekirjoitetaan XML-merkkijono, joka sisältää kriittistä tietoa jokaisesta tilauksesta. XML-merkkijono on listauksessa 4.4.

```
<?xml version="1.0" encoding="utf-16"?>
<ArrayOfTracerOrderSignature  xmlns:xsi="http://www.w3.org/2001/XMLSchema-in-
stance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <TracerOrderSignature>
    <TracerName>[18F]FDG</TracerName>
    <Radioactivity>2694</Radioactivity>
    <OrderCode>3cc2e8c7-cac9-4a8f-a5aa-1b4faaa14034</OrderCode>
  </TracerOrderSignature>
  <TracerOrderSignature>
    <TracerName>[68Ga]DOTANOC</TracerName>
    <Radioactivity>175</Radioactivity>
    <OrderCode>60a454a8-7a65-4cb3-8971-a3e36bae4dde</OrderCode>
  </TracerOrderSignature>
</ArrayOfTracerOrderSignature>
```

Listaus 4.4: Merkkiaineiden hyväksynnässä allekirjoitettava XML-tiedosto.

Jokaiselle tilaukselle allekirjoitetaan merkkiaineen nimi, tilattu aktiivisuus ja UUID, joka yhdistää merkkiaineen PETERP-tietokannassa olevaan olioon. Taulukko 6 esittää luokan sisältämän tiedon.

Taulukko 6: Hyväksyttejen merkkiainetilausten allekirjoitukset.

Attribuutti	Tyyppi	Kuvaus	Allekirjoitetaan
SignerName	NVARCHAR(MAX)	Allekirjoittajan nimi	Ei
SignDate	DateTime	Allekirjoitusaika	Kyllä
XMLSerializedData	NVARCHAR(MAX)	XML-merkkijono, joka sisältää allekirjoitettavan tiedon.	Kyllä

Allekirjoitettavaa tietoa on XML-merkkijono ja allekirjoituspäivämäärä. Allekirjoittajan nimi metatietoa, jonka avulla tietokantaan voi tehdä hakuja.

Valmistusprotokollat: BatchApproval-taulu

Valmistusprotokollien allekirjoituksessa käytetty BatchApproval-luokka sisältää taulukon 7 mukaiset tiedot.

Taulukko 7: BatchApproval on protokollien hyväksymiseen käytetty luokka.

Attribuutti	Tyyppi	Kuvaus	Allekirjoitetaan
SubstanceCode	NVARCHAR(MAX)	Allekirjoitettavan merkkiaineen tai pullo-ohjeen nimi. Esimerkiksi ”[18F]FDG”	Ei
AuditNumber	Integer32	Allekirjoitettavan merkkiaineen tai pullo-ohjeen versionumero. Nimi tulee suoraan LIMS toteutuksen tietokannasta, jossa versionumeroa kutsutaan nimellä AuditNumber.	Ei
SignedXML	NVARCHAR(MAX)	Allekirjoitettava XML-merkkijono.	Kyllä
SignerName	NVARCHAR(MAX)	Allekirjoittajan nimi.	Ei

Valmistusprotokollien hyväksyntä -allekirjoituksessa vain SignedXML-kenttä allekirjoitetaan. Muut kentät ovat metatietoa, jonka avulla voi tehdä hakuja tietokannasta. Valmistusprotokollille hakuja tehdään SubstanceCoden ja AuditNumberin perusteella ja haun tuloksena on SignerName. SignerName on henkilö, joka on suorittanut allekirjoituksen. Kantaluokasta on periytetty kolme luokkaa (*Compiler*, *Inspector* ja *Acceptor*), jotka kertovat missä roolissa allekirjoitus on annettu. Allekirjoitettava XML-merkkijono on listauksen 4.5 mukainen.

```
<?xml version="1.0" encoding="utf-16"?>
<Substance              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Tracer>
    <Name>[18F]FDG</Name>
    <Isotope>18F</Isotope>
  </Tracer>
  <SubstanceTests>
    <SubstanceTest>
      <TestDescription>Measure radiation</TestDescription>
      <InstrumentCode>PET 232 VDV-505</InstrumentCode>
      <GroupCode>Endpr_EndProduct</GroupCode>
      <TestCode>MeasureRad</TestCode>
      <TestCodeVersion>4</TestCodeVersion>
      <StepType>BatchInstruction</StepType>
    </SubstanceTest>
    <SubstanceTest>
      <TestDescription>Merkkaa synteesi tehdyksi</TestDescription>
      <GroupCode>Endpr_EndProduct</GroupCode>
      <TestCode>Synteesi</TestCode>
      <TestCodeVersion>1</TestCodeVersion>
      <StepType>BatchInstruction</StepType>
    </SubstanceTest>
  </SubstanceTests>
  <QAReleaseRequired>true</QAReleaseRequired>
  <Description>Protokolla allekirjoitusten testaamiseen</Description>
  <SubstanceCode>SignatureTests</SubstanceCode>
</Substance>
```

Listaus 4.5: Protokollien allekirjoituksessa allekirjoitettava XML-tiedosto.

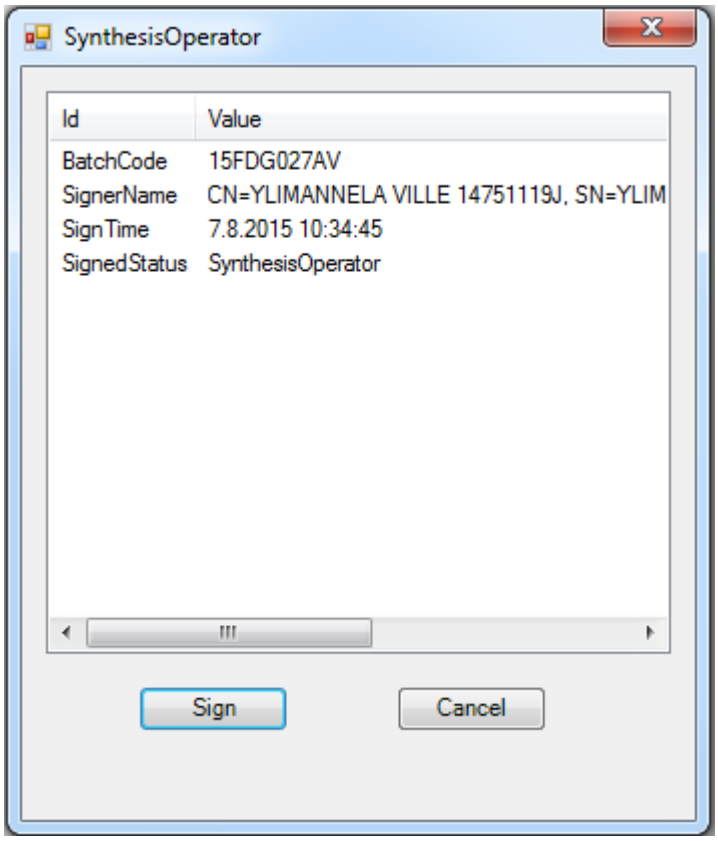
Valmistusprotokollien allekirjoituksessa allekirjoitettava XML-merkkijono sisältää allekirjoitettavan merkkiaineen, siihen liittyvät testit, valmistusprotokollan kuvauksen ja valmistusprotokollan nimen.

4.3 Allekirjoitettavan datan visualisointi

Jokainen allekirjoitus visualisoidaan käyttäjälle. Visualisoitava data vaihtelee allekirjoitustyyppien välillä. Allekirjoitukseen on kaksi eri käyttöliittymää:

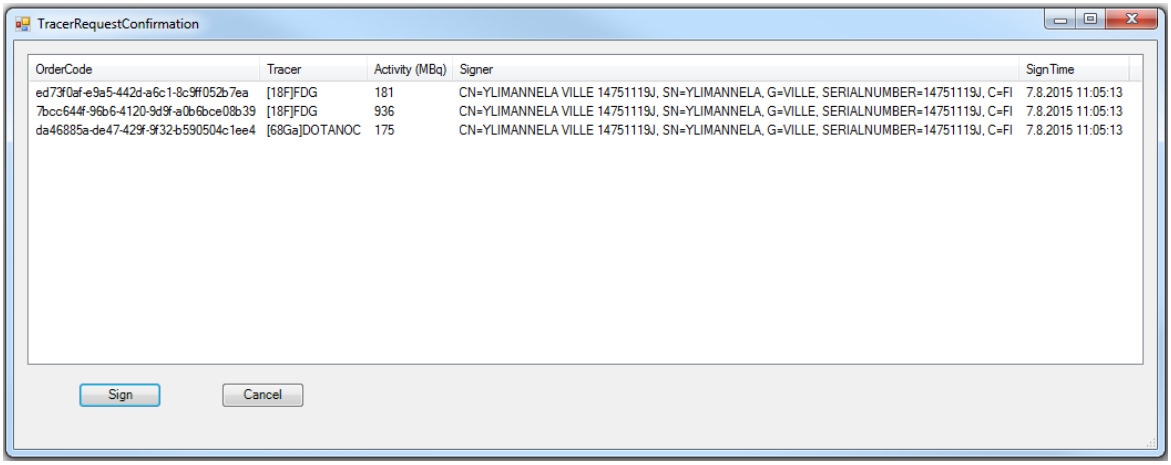
- kuittauksille ja tilausten hyväksynnälle, sekä
- protokollien hyväksynnälle.

Kuittaukset ja tilausten hyväksyntä allekirjoitetaan käyttämällä samaa käyttöliittymäkomponenttia. Tuotannon allekirjoituksien yhteydessä käyttäjälle näytetään allekirjoitettava tieto: merkkiaineen eräkoodi, allekirjoittajan nimi, allekirjoitusaika ja allekirjoitettava toiminto. Kuva 10 esittää kuittauksissa käytettävän käyttöliittymän ja siinä käyttäjälle näkyvän tiedon.



Kuva 10: Kuittauksien käyttöliittymä.

Allekirjoituskomponentti esittää tilausten hyväksynnän yhteydessä käyttäjälle matriisin allekirjoitettavasta tiedosta. Käyttäjälle ei esitetä allekirjoitettavaa XML-merkkijonoa, vaan informaatio visualisoidaan käyttöliittymäkomponentissa. Käyttäjä kuitenkin näkee kaiken tarvittavan tiedon: merkkiaineen nimen, merkkiaineen yksilöivän uniikin tunnisteen, tilatun aktiivisuuden määrän, allekirjoittajan nimen ja allekirjoitusajan. Kuvassa 11 esitellään allekirjoituskomponentti, kun käyttäjä allekirjoittaa kolme eri merkkiainetilausta: 2 FDG-tilausta ja 1 DOTANOC-tilaus.



Kuva 11: Merkkiainetilauksen hyväksyntä.

Valmistusprotokollien hyväksynnälle on oma allekirjoitusnäkymänsä. Tässä allekirjoitusnäkymässä käyttäjälle näytetään protokollalle aikaisemmin annetut allekirjoitukset. Valmistusprotokollien allekirjoitukset ovat sidottu kahteen asiaan: valmistusprotokollan nimeen ja versioon. Version kasvaessa kaikki neljä allekirjoitusta tulee antaa uudelleen. Versio kasvaa, kun valmistusprotokollaan tehdään muutos. Allekirjoitettavaa dataa ei käyttäjälle visualisoida. Kuva 12 on esimerkki protokollan hyväksyttämisessä käytettävästä käyttöliittymästä.

Kuva 12: Protokollien allekirjoitusnäkymä.

Protokollien tapauksessa käyttäjälle näytetään vain hyväksyttävä protokolla ja sen versio. Käyttäjä valitsee itse allekirjoittamansa pisteen. Käyttöliittymä ei anna allekirjoittaa samaa tietoa useaan kertaan. Jokaisen allekirjoituksen jälkeen Sign-napin tilalle ilmestyy

allekirjoituksen antaneen varmenteen Subject-kentän arvo, joka on esitelty luvussa 3.2.3. Allekirjoitukset on annettava oikeassa järjestyksessä. Käyttöliittymässä oikea etenemisjärjestys on ylhäältä alas.

5. TOTEUTUKSEN RIITTÄVYYDEN ARVIOINTI

Luvussa 5 tehdään yhteenveto alkuperäisestä toteutuksesta löydetyistä puutteista. Puutteista osa valitaan korjattavaksi ja tehdyt korjaukset esitellään luvussa 6. Toteutusta arvioidaan neljästä eri näkökulmasta: lainsäädäntö, tekninen toteutus, allekirjoituksien sisältö ja visualisointi.

Osa näkökulmista on helpommin validoitavissa; *Laki vahvasta sähköisestä tunnistamisesta ja sähköisistä allekirjoituksista* kertoo selvästi mitä digitaaliselta allekirjoitukselta vaaditaan. Toinen ääripää on visualisointi, jonka virheitä on hankala löytää. Digitaalisten allekirjoituksien visualisoinnille ei ole standardeja, joihin alkuperäistä toteutusta voi verrata.

5.1 Lainsäädännön vaatimusten täyttäminen

Luvussa 3.1.2 esiteltiin *laki vahvasta sähköisestä tunnistamisesta ja sähköisistä allekirjoituksista*. Laissa termille *kehittynyt sähköinen allekirjoitus* asetetaan neljä vaatimusta. Vaatimukset ja niiden toteutuneisuus on esitetty taulukossa 8.

Taulukko 8: Suomen lainsäädännön asettamat vaatimukset kehittyneelle sähköiselle allekirjoitukselle (Laki vahvasta sähköisestä tunnistamisesta ja sähköisistä allekirjoituksista 7.8.2009/617, 2§).

Vaatimus	**Toteutuneisuus**	**Perustelut**
Liittyy yksiselitteisesti allekirjoittajaan.	Kyllä	Käytössä on julkisen avaimen kryptografia (RSA), joka on yleisesti käytetty allekirjoitusalgoritmi. Käyttäjien varmenteet generoidaan luotettavan tahon toimesta (VRK).
Voi yksilöidä allekirjoittajan	Kyllä	Jokaisella käyttäjällä on yksilöivä varmenne, joka linkittää julkisen avaimen avulla käyttäjän ja allekirjoituksen.
Allekirjoittaja pitää menetelmän yksinomaisessa valvonnassa.	Kyllä	Allekirjoitukset tehdään VRK:n myöntämällä henkilökohtaisella älykortilla. Allekirjoittaessa käyttäjän tulee vahvistaa allekirjoitus henkilökohtaisella PIN-koodilla.
Liitetty sähköiseen tietoon siten, että muutokset voidaan havaita.	Kyllä	Allekirjoitettu data jää talteen tietokantaan eikä tietokannan tieto ole muokattavissa www-sovelluspalvelimen yli tehtävien rajapintakutsujen kautta.

Luvussa 4 esitelty alkuperäinen toteutus täyttää kaikki taulukon 8 vaatimukset. Vaatimukset eivät ota kantaa tekniseen toteutukseen tai allekirjoitettavan tiedon sisältöön, vaan vaatimukset ovat hyviä lähtöperiaatteita digitaaliselle allekirjoitukselle.

Allekirjoituksen luomisväline asettaa vaatimuksia laitteistolle jolla allekirjoitus luodaan. *Laki vahvasta sähköisestä tunnistamisesta ja sähköisistä allekirjoituksista* vaatii luomisvälineeltä alla olevan taulukon mukaiset ominaisuudet.

Taulukko 9: Suomen lainsäädännön asettamat vaatimukset allekirjoituksen luomisvälineelle (Laki vahvasta sähköisestä tunnistamisesta ja sähköisistä allekirjoituksista 7.8.2009/617, 28 §).

Vaatimus	**Toteutuneisuus**	**Perustelut**
Allekirjoituksen luomistiedot ovat ainutlaatuisia ja luottamuksellisia.	Kyllä	Jokaisella käyttäjällä on oma yksityinen avain, joka sijaitsee turvallisesti älykortilla.
Allekirjoituksen luomistietoja ei voi päätellä muista tiedoista.	Kyllä	Käytettyä yksityistä avainta ei ole mahdollista päätellä allekirjoituksesta tai julkisesta avaimesta.
Allekirjoitus on suojattu väärentämiseltä.	Kyllä	Käytössä on laajasti käytetty RSA-algoritmi. Yksityinen avain on turvassa älykortilla. Muutokset allekirjoitettuun tietoon huomataan tiivisteen muutoksesta.
Allekirjoittaja voi suojata allekirjoituksen luomistiedot muiden käytöltä.	Kyllä	Älykortit kulkevat henkilöiden mukana. Älykortilla sijaitseva yksityinen avain on suojattu PIN-koodilla.
Luomisväline ei muuta allekirjoitettavia tietoja.	Kyllä	Allekirjoitettava tieto on tiivistetty. Allekirjoitus tehdään asiakasohjelmistossa ja allekirjoituksen oikeellisuus varmistetaan ennen tallentamista tietokantaan myös palvelimella.

Luvussa 3.1.2 esitetyn tulkinnan mukaan *allekirjoituksen luomisväline* on älykortti, kortinlukija ja näihin liittyvät ajurit. Luomisväline voi tästä johtuen vaihdella työasemakohtaisesti, esimerkiksi erilaisen kortinlukijan takia. Tämän diplomityön tarkoitus on kuitenkin vahvistaa, että tehty toteutus on riittävä. Luomisvälineet ovat 3. osapuolen komponentteja joiden validointi on työn ulkopuolella.

5.2 Teknisessä toteutuksessa havaitut puutteet

Projektissa tehdyn teknisen toteutuksen sisältöä verrataan luvussa 3 esiteltyihin XML- ja BDOC-allekirjoitusstandardeihin. Lainsäädäntö tai GMP-ohjeistus eivät aseta vaatimuksia tekniselle toteutukselle, kuten käytetyille algoritmeille tai sarjallistamiselle. Tekninen näkökulma on kuitenkin tärkeää huomioida, sillä huono toteutus voi aiheuttaa ongelmia järjestelmää jatkokehitettäessä tai kriittisen tiedon puute voi aiheuttaa tulevaisuudessa ongelmia allekirjoituksia validoitaessa.

Seuraavat puutteet ovat tulleet esille, kun toteutusta on testattu, kehitetty ja verrattu luvussa 3.3.1 esiteltyyn XML-allekirjoitusstandardiin:

1. Allekirjoitukset eivät sisällä kaikkea tarpeellista tietoa mitä vaaditaan allekirjoituksen vahvistamiseen kolmannen osapuolen toimesta, kuten esimerkiksi käytetyt algoritmit.
2. Allekirjoituksessa käytettyjä algoritmeja ei allekirjoiteta.
3. Ei ole aina selvää mitä dataa allekirjoitetaan. Vaikka tieto on perityssä luokassa, ei se tarkoita että sitä allekirjoitetaan.
4. Allekirjoittamisessa käytettävästä sarjallistamisfunktiosta löytyi vika, jonka johdosto ajat (C# DateTime-olio) eivät sarjallistu aina samanlaiseksi tavujonoksi. Vian johdosta allekirjoituksen vahvistaminen, käyttämällä tietokannasta haettua dataa, ei onnistu.
5. Sarjallistaminen ei tapahdu standardilla tavalla. Tällä tarkoitetaan että muunto merkkijonosta (C# String-olio) binääriksi ei tapahdu tavalla, joka olisi yleisesti tunnettu ja hyväksytty tapa.
6. Nykyinen luvussa 4.1.5 esitelty sarjallistamisfunktio sarjallistaa koko olion, ei pelkästään allekirjoituksen kannalta merkittävää dataa. Tämä voi aiheuttaa ongelmia tilanteissa joissa alkuperäisen olion luominen ei onnistu. Koko olion sarjallistaminen myös tarkoittaa, että allekirjoituksien vahvistaminen ei tule onnistumaan muissa ohjelmointiympäristöissä.
7. Palvelimella allekirjoituksen vahvistus tapahtuu käyttämällä asiakkaalta tulevaa dataa. Alkuperäinen toteutus vahvistaa että asiakkaan allekirjoittama data ei ole muuttunut. Alkuperäinen toteutus **ei** vahvista että allekirjoitettu data on oikeaa. Esimerkiksi merkkiainetilausten vahvistuksen yhteydessä tiedetään että palvelinkutsun parametrina tuleva XML on allekirjoitettu, mutta ei vahvisteta että allekirjoitettu XML sisältää hyväksytyn merkkiaine-erän.
8. Käytössä on MD5-algoritmi. MD5 on nykyään murrettavissa. Tiivistäminen MD5:llä sarjallistamisen yhteydessä on turha vaihe, joka vain heikentää allekirjoituksen vahvuutta.

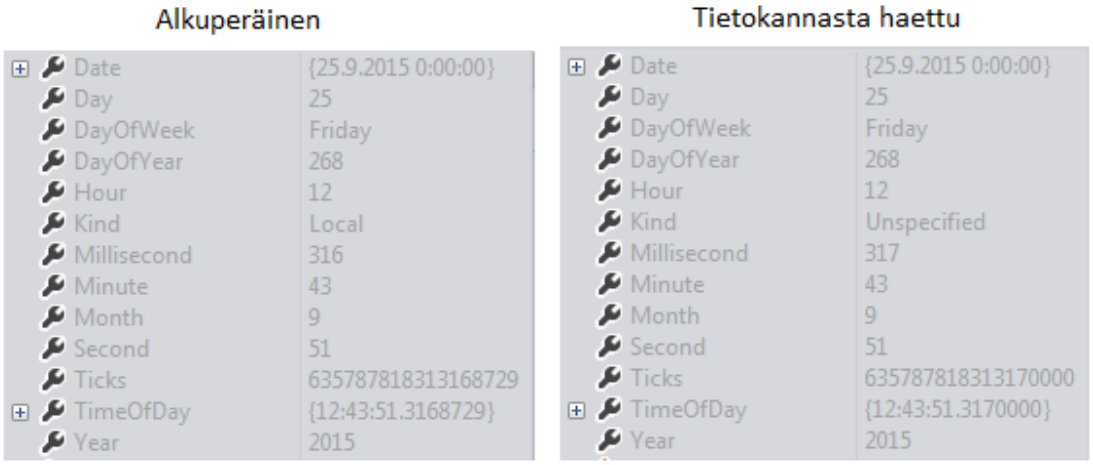
Puutteista kriittisimpiä ovat kriittisten tietojen puute allekirjoituksesta, MD5-algoritmi ja ongelmat sarjallistamisessa.

5.2.1 Sarjallistamisen ongelmat

Allekirjoitettavan datan sarjallistaminen binääriksi ei tapahdu standardilla tavalla, joka voi johtaa ongelmiin tulevaisuudessa. Sarjallistamisessa on jo havaittu ongelmia DateTime-olioilla.

Sarjallistamisfunktion heikkous ja vahvuus on sen joustavuus. Toteutuksen avulla voi sarjallistaa muitakin kuin String, DateTime tai Int -tyyppiä olevia olioita. Ongelmia voi syntyä, jos sarjallistettavan olion binäärimuoto riippuu ajasta jolloin se on luotu tai satunnaisuudesta. Kuvitellaan tilanne, jossa olion rakentajassa asetetaan jäsenmuuttujaan olion luontiaika tai oliolle luodaan oma, globaalisti yksilöivä UUID (Universally Unique Identifier). Kuvitteellisessa tilanteessa luotu olio tulee aina sarjallistumaan erilaiseksi tavujonoksi ja tuottamaan eri tiivisteen. Olion sarjallistuminen erilaiseksi tavujonoksi tarkoittaa, että allekirjoitusta ei voi vahvistaa jälkeenpäin, sillä uusi instanssi oliosta ei tule vastaamaan allekirjoitettua oliota. Lopputuloksena on mitätön allekirjoitus.

Ongelma syntyy, kun tietokannasta luettava olio poikkeaa sinne tallennetusta oliosta. Kuva 13 on otettu Visual Studio 2012 debuggerista. Kuvassa vasemmalla on alkuperäinen tietokantaan tallennettava DateTime-olio. Oikealla kuvassa on tietokannasta haettu DateTime-olio. Alkuperäisessä oliossa Ticks-arvo on tarkempi; kannasta haettaessa arvo on pyöristetty ylöspäin lähimpään kymmeneen tuhanteen. Ticks-arvo esittää kulunutta aikaa päivämäärästä 12:00 01.01.0001. Yksi ”Tick” on sekunnin 10 miljoonasosa. Toinen muuttunut jäsenmuuttuja on Kind-arvo, joka kertoo onko aika lokaalia aikaa, UTC aikaa vai määrittelemätön. [23][24]



Kuva 13: Tallennettu ja tietokannasta haetut oliot eroavat toisistaan. Kentät Millisecond, Ticks, Kind ja TimeOfDay ovat erilaisia.

Alkuperäisen ja tietokannasta luettavan olion erotessa ne sarjallistuvat erilaiseksi tavujonoksi. Allekirjoitusta ei voi vahvistaa sen jälkeen, kun alkuperäinen olio katoaa muistista. Koska allekirjoitusten vahvistus tehdään palvelimella samasta oliosta kuin asiakasohjelmistossa, ei ongelmaa ole aiemmin huomattu. Ongelma on kriittinen ja se on korjattava.

5.3 Allekirjoituksien sisältö

Allekirjoituksen sisältöön vaatimuksia tulee kahdesta lähteestä: GMP-vaatimuksista ja asiakkaalta saadusta dokumentaatiosta. Luvussa 3.1.1 esitellyn GMP-standardin asettamat vaatimukset ovat taulukossa 10.

Taulukko 10: GMP-standardin asettamat vaatimukset.

Vaatimus	**Toteutuneisuus**	**Perustelut**
Elektronisia tallenteita voidaan tallentaa digitaalisesti.	Kyllä	Digitaaliset allekirjoitukset ovat käytössä järjestelmässä.
Digitaalisilla allekirjoituksilla on oltava sama vaikutus kuin fyysisillä.	Kyllä	Uudessa järjestelmässä digitaaliset allekirjoitukset tulevat korvaamaan fyysiset allekirjoitukset. Digitaaliset allekirjoitukset annetaan samoissa prosessin pisteissä kuin fyysiset.
Allekirjoitukset pitää linkittää pysyvästi allekirjoitettuun tietoon.	Kyllä	Allekirjoitettu data jää talteen tietokantaan eikä tietokannan tieto ole muokattavissa www-sovelluspalvelimen kautta.
Allekirjoitusten tulee sisältää allekirjoituksen aika ja päivämäärä.	Ei	Allekirjoitusaika puuttuu protokollanhyväksyntäallekirjoituksista.
Tuote-erien vapautuksessa pitää käyttää digitaalisia allekirjoituksia. Vain sallituilla henkilöillä on oikeus vapauttaa tuote-eriä ja järjestelmän tulee tunnistaa käyttäjät ja tallentaa vapauttajan tunnistetiedot.	Kyllä	Tuote erien käyttöönvapautus tapahtuu merkkiainetuotantoprosessin allekirjoituksella. Allekirjoitus voidaan linkittää varmenteeseen ja varmenne voidaan linkittää käyttäjään. Allekirjoittaminen (ml. käyttöönvapautus) on mahdollista vain varmenteilla jotka ovat lisätty järjestelmään pääkäyttäjän tunnuksilla.

Alkuperäinen toteutus ei täytä GMP-standardin asettamia vaatimuksia, koska kaikissa allekirjoituksissa ei ole allekirjoituspäivämäärää. Luvun 3.1.1 kohdassa d esitetyn tulkinnan mukaan aikaleima allekirjoituksen metatiedossa ei riitä, vaan päivämäärä on allekirjoitettavaa tietoa. Kaikkien GMP-standardin asettamien vaatimusten tarvitsee täyttyä, joten tieto on lisättävä allekirjoituksiin.

Luvun 3.1.1 kohdan e mukaan GMP-standardi ei suoraan vaadi allekirjoittajan nimen sisällyttämistä allekirjoitukseen, mutta käyttöönvapautuksen yhteydessä tulee tallentaa käyttöönvapauttajan tunnistetiedot. Tunnistetiedot ovat helposti lisättävissä allekirjoitukseen, joka olisi järkevää ongelmien välttämiseksi tulevaisuudessa.

Allekirjoitusten toteutuksessa on havaittu seuraavia puutteita vertaamalla toteutettuja allekirjoituksia nykyisiin merkkiaineohjeisiin ja merkkiaine-erien pöytäkirjoihin:

1. Allekirjoittajan rooli puuttuu valmistusprotokollien allekirjoituksista. Rooli on kriittistä tietoa, sillä jokaisella allekirjoituksella on eri tarkoitus.
2. Valmistusprotokollien allekirjoituksissa allekirjoitettava XML on vajavainen. Puuttuvaa tietoa ovat esimerkiksi mittausten raja-arvot, jotka voivat olla allekirjoituksen sisällön kannalta tärkeitä.
3. Merkkiainetilauksen tuotantoon hyväksyntään käytettävät allekirjoitukset eivät sisällä injektioita. Injektiot ovat kuitenkin tietoa, joka on prosessin suunnittelun kannalta erittäin kriittistä, sillä injektioiden aika ja pyydetty määrä vaikuttavat merkittävästi tehtävän merkkiaineen määrä.
4. Fyysisessä paperissa synteesin allekirjoittaja vakuuttaa eräpöytäkirjassa ”End Product” -vaiheessa olevien tietojen paikkansapitävyyden. Tällainen tieto on esimerkiksi aktiivisuus. Tämä ei ole tällä hetkellä allekirjoitettavaa tietoa.

Merkkiainetuotantoprosessin allekirjoitusten merkittävin ongelma on tiedon vähyys. Allekirjoitettava tieto ei tällä hetkellä sisällä mitään tietoa jota merkkiainepöytäkirjaan on täytetty. Ongelmatilanteissa joissa pitää jälkeenpäin tarkistaa hyväksytty data, ei tarkistus onnistu. Tämä johtuu siitä, että itse synteesin dataa ei allekirjoiteta, vaan allekirjoittaja allekirjoituksellaan hyväksyy sen että prosessi voi edetä. Kaikki merkkiainepöytäkirjaan täytetty tieto ja tietoon muutoshistoria tallentuvat tietokantaan. Merkkiainepöytäkirjaan täytetyt tiedot ovat nähtävissä myös jälkeenpäin, mutta tietoa ei ole allekirjoitettu.

Merkkiainetilausten hyväksyntä -allekirjoituksen suurin puute on metatiedon puute. Yhdessä XML-tiedostossa voi olla useamman merkkiainetilauksen hyväksyntä. Tällä hetkellä ei ole mahdollista tehdä nopeita tietokantahakuja merkkiaine-erän UUID-tunnisteen perusteella. Haku voi olla tarpeellinen, jos halutaan etsiä merkkiaine-erän tuotantoon hyväksyneen käyttäjän nimi. Koska hyväksyttyjä merkkiaineita on n-kappaletta yhtä allekirjoitusta kohti, ei kyseistä tietoa voi sisällyttää yhteen tauluun. Hyväksytyille tilauksille on luotava oma taulunsa, mikäli tiedon avulla on tarpeellista tehdä hakuja.

Merkkiainetilausten hyväksyntä -allekirjoituksissa on hyvä tietää myös ensimmäinen injektio. Ensimmäisen injektion aika ratkaisee, milloin erän tarvitsee olla valmis.

Valmistusprotokollan allekirjoituksista puuttuu kahta tietoa: tietoa protokollasta ja allekirjoittajan rooli. Listauksen 4.5 XML ei sisällä kaikkea tietoa protokollista. Puuttuvaa tietoa ovat solut, joihin käyttäjä voi listata arvoja ja näihin liittyvät tiedot, kuten sallitut mittausten raja-arvot, yksiköt ja pyöristykset. Puuttuvaa tietoa on myös rooli, jossa allekirjoittaja on allekirjoituksensa antanut. Alkuperäisessä toteutuksessa rooli ei kuulu allekirjoitukseen, vaan se on vierasavainviite tietokannassa. Tieto voi olla jossain tilanteissa

riittämätön, sillä fyysisestä paperissa allekirjoituksen paikka vaikuttaa allekirjoituksen sisältöön; jos allekirjoitus on ”Approved by”-rivillä, on allekirjoituksen antaja hyväksynyt uuden valmistusprotokollan käyttöön.

5.4 Allekirjoitettavan tiedon visualisointi

Visualisoinnista olevia puutteita on vaikeampi arvioida kuin teknisiä vaatimuksia. Suomen laki tai GMP-standardi eivät aseta visualisointivaatimuksia, joten toteutusta ei voi suoraan peilata olemassa oleviin vaatimuksiin.

Visualisoinnissa on havaittu seuraavat puutteet:

1. Digitaalisille allekirjoituksille ei ole katselutyökalua. Allekirjoitukset luodaan asiakasohjelmistossa ja tarkastetaan palvelimella. Tallennuksen jälkeen allekirjoituksia ei kuitenkaan voi katsella jälkeenpäin.
2. Kaikkea allekirjoitettavaa dataa ei näytetä käyttäjälle protokollien allekirjoitusten yhteydessä. Kuva 12 esittää kaiken allekirjoittajalle visualisoitavan tiedon.
3. Merkkiainetuotantoprosessin allekirjoituksissa ei ilmaista riittävän selvästi mihin käyttäjä on laittamassa allekirjoituksensa ja mitä allekirjoituksen antaminen tarkoittaa. Kuva 10 esittää käyttäjälle näytettävän tiedon.

Alkuperäinen toteutus ei täytä yllä olevia kohtia. Kaikki kohdat ovat kuitenkin visualisoinnin kannalta tärkeitä.

6. KORJATTU TOTEUTUS

Luvussa tarkastellaan edellisessä luvussa tunnistettuja puutteita ja esitellään korjattu toteutus. Osa puutteista todettiin työn aikana niin vakaviksi laboratorion toiminnan kannalta, että korjaustyöhön päätettiin ryhtyä työn kirjoituksen aikana. Digitaaliset allekirjoitukset ovat tässä luvussa esiteltyjen korjausten jälkeen riittäviä lain ja säädösten, teknisen toteutuksen sekä sisällön puolesta. Korjaukset jaetaan kolmeen kategoriaan: teknisen toteutuksen korjaus, tietomallin korjaus ja allekirjoitettavan tiedon täydentäminen. Lopuksi luvussa esitellään korjatun toteutuksen testaukseen vaaditut askeleet.

Teknisen toteutuksen korjaus keskittyy sarjallistamisen, tiivistämisen ja salauksen korjaukseen. Sarjallistamisessa on havaittu ongelmia jotka liittyvät tiedon tallentamiseen tietokantaan. Allekirjoitettava tieto ajetaan tarpeettomasti kahden tiivistefunktion lävitse.

Tietomallin korjauksessa tietomalliin lisätään puuttuvaa tietoa, poistetaan redundanttia tietoa ja muuttujien nimiä yhtenäistetään. Puuttuvaa tietoa ovat käytetyt algoritmit. Tietokantaan ei jää merkintää käytetyistä sarjallistamis-, tiivistys- ja allekirjoitusalgoritmeista. Tiedon puute tulee aiheuttamaan ongelmia, mikäli jokin edellä mainituista vaihtuu. Ongelmat johtuvat siitä, että allekirjoitukset ovat vahvistettavissa vain käyttämällä samoja algoritmeja kuin allekirjoituksessa. Jos allekirjoituksissa ei ole tietoa millä algoritmeilla ne on luotu, ei niitä voi jälkeenpäin vahvistaa.

Allekirjoitettavassa tiedossa on havaittu puutteita. Puuttuvaa tietoa ovat esimerkiksi allekirjoittajan nimi ja allekirjoitusaika. Nämä tullaan lisäämään jokaiseen allekirjoitukseen.

6.1 Teknisen toteutuksen korjaus

Teknisessä toteutuksessa havaittiin kaksi ongelmaa: sarjallistamisfunktio ei toiminut oikein DateTime-olioilla ja allekirjoitettava data tiivistettiin kahdesti eri tiivistefunktioilla. MD5-tiivistys jätettiin pois korjatusta toteutuksesta. Uusi käytettävä tiivistysalgoritmi on SHA-1.

Allekirjoituksissa on tarpeetonta allekirjoittaa muita kuin merkkijonoja, sillä kaikki allekirjoitettava informaatio on sarjallistettavissa merkkijonoiksi. Esimerkiksi XML-merkkijonon avulla voi helposti esittää rakenteellista tietoa. Tehtiin päätös luopua toteutuksesta, joka sarjallistaa useita eri tyyppiä olevia olioita. Uusi toteutus sarjallistaa vain merkkijonoja. Kaikki allekirjoitettava tieto sarjallistetaan korjatussa toteutuksessa XML-merkkijonoksi, josta kerrotaan lisää luvussa 6.3. Listaus 6.1 esittää toteutuksen, joka korvaa listauksessa 4.1 esitetyn *CalculateHash*-funktion.

```
public static byte[] SerializeUnicodeString(string stringToSign)
{
      UnicodeEncoding encoder = new UnicodeEncoding();
      return encoder.GetBytes(stringToSign);
}
```

Listaus 6.1: Uusi sarjallistamisfunktio. Funktio sarjallistaa Unicode-koodatun merkkijonon tavujonoksi.

Uusi toteutus sarjallistaa merkkijonon käyttämällä .NET-komponenttikirjaston valmista toteutusta. Uusi toteutus palauttaa sarjallistetun, mutta ei-tiivistetyn, merkkijonon. Tästä syystä luotiin listauksen 6.2 mukainen toteutus SHA-1-tiivisteen laskemiseksi. Sille annetaan parametriksi listauksen 6.1 tuottama tavujono.

```
public static byte[] CalculateSHA1(byte[] bytesToHash)
{
      var sha1csp = new SHA1CryptoServiceProvider();
      return sha1csp.ComputeHash(bytesToHash);
}
```

Listaus 6.2: Funktio laskee SHA1-tiivisteen annetusta tavujonosta.

CalulateSHA1-funktion avulla lasketaan *DigitalSignature*-kantaluokan *Hash*-jäsenmuuttujan arvo. *Hash*-arvosta on apua tilanteissa joissa allekirjoituksen vahvistaminen epäonnistuu. *Hash*-arvon avulla voidaan selvittää, onko allekirjoitus vai allekirjoitettu tieto muuttunut. Jos *SignedXML*-jäsenmuuttjan tiiviste vastaa *Hash*-jäsenmuuttujan arvoa, on muutos allekirjoituksessa. Jos arvot eivät vastaa toisiaan, on allekirjoitettu data muuttunut.

Vanha sarjallistamis- ja tiivistysfunktio jätettiin entiselleen, mutta uudelleennimettiin. Uusi nimi on *CalculateHashVersion1*. Vanhaa funktiota tarvitaan edelleen jo olemassa olevien PETERP-allekirjoitusten vahvistamiseksi. Kyseinen funktio merkattiin attribuutilla ”Obsolete”. Attribuutti kehottaa ohjelmoijaa käyttämään uudempaa SerializeUnicodeString -funktiota, mutta ei estä vanhan käyttämistä. Attribuutti alleviivaa sillä merkatun funktion kuvan 15 mukaisesti.

6.2 Tietomallin korjaus

Tietomallissa huomattiin useita puutteita työn aikana. Jokaiseen luokkaan tehtiin muutoksia. Syyt muutoksille ovat tarvittavien tietojen puute, epäloogiset perintähierarkiat ja muutokset allekirjoitettavan tiedon tyyppiin. Korjatun tietomallin saavutetut edut ovat:

1. helpompi jatkokehitettävyys,
2. allekirjoituksien löytäminen tiekannasta metatiedon perusteella,
3. yhtenäisempi allekirjoitusten rakenne, sekä
4. turhien tietorakenteiden poisto.

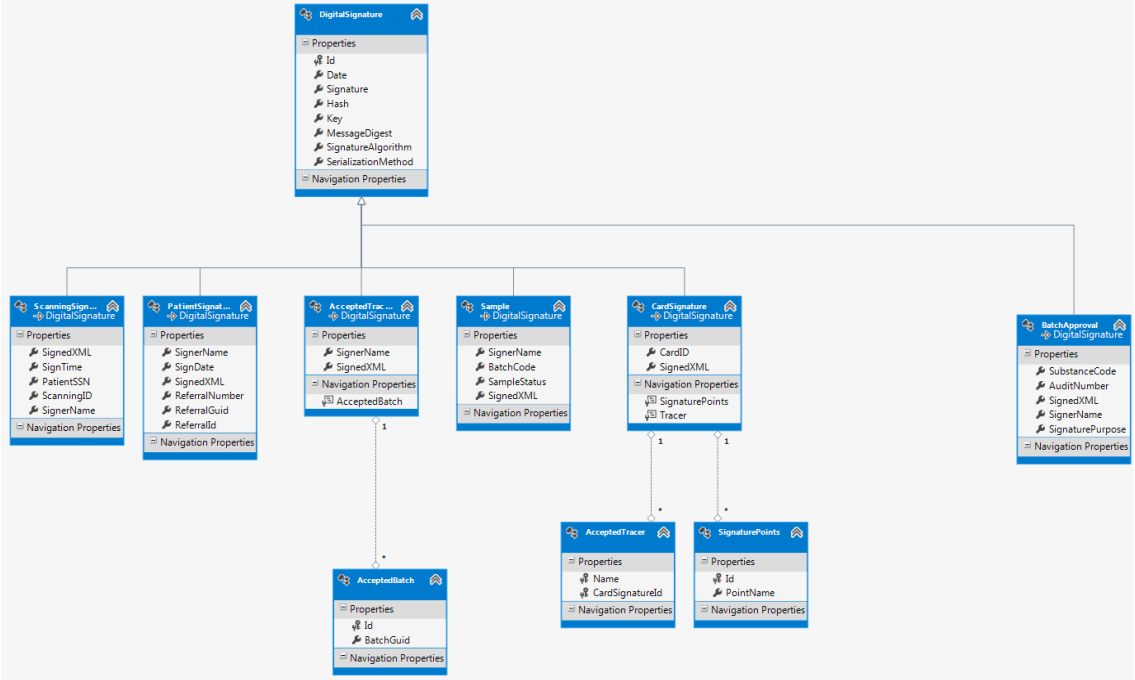
DigitalSignature-kantaluokkaan on lisätty allekirjoituksesta puuttuvat tiivistys-, allekirjoitus- ja sarjallistamisfunktiot. Kaikki tiedot ovat String-tyyppisiä olioita, jotka tallennetaan tietokantaan NVARCHAR-tyyppeinä. Kentät voivat tällä hetkellä saada seuraavat arvot:

- MessageDigest: [MD5 | SHA1CryptoServiceProvider]. Arvolla viitataan .NET-luokkaan mitä tiivisteen luomiseen on käytetty.
- SignatureAlgorithm: [RSACryptoServiceProvider]. Allekirjoituksessa käytetty allekirjoitusalgoritmi.
- SerializationMethod: [MemoryStream+BinaryFormatter v1 | UnicodeEncoding.GetBytes(string)]. Vanhat allekirjoitukset ovat sarjallistettu MemoryStream+BinaryFormatter -funktioilla ja uudet korjatun toteutuksen allekirjoitukset Encoding.GetBytes.funktiolla.

AcceptedTracerOrder-luokan allekirjoitettava XML on nimetty uudelleen. Uusi nimi on SignedXML, joka noudattaa samaa nimeämistapaa kuin muidenkin allekirjoitusten nimeäminen. AcceptedTracerOrder-luokkaan on lisätty assosiaatio uuteen Acceptedbatch-luokkaan. AcceptedBatch-luokassa on tarvittava metatieto, jonka avulla tietokannasta voidaan etsiä merkkiaine-erään liittyvän allekirjoitus. Koska yksi allekirjoitus voi sisältää useamman hyväksytyn merkkiainetilauksen, tarvitaan yhdestä-moneen -assosiaatio. Tästä syystä AcceptedBatch on oma luokkansa.

Sample-luokassa on siirrytty allekirjoittamaan XML-merkkijonoa kuten muissakin allekirjoituspisteissä. Muutos helpottaa allekirjoitettavan datan sisällön muokkausta tulevaisuudessa. Muilta osin luokka on pysynyt entisellään.

BatchApproval-luokkaa on yksinkertaistettu. Kaikki allekirjoitettava data on edelleen SignedXML-merkkijonossa. Tästä perityt kolme luokkaa on poistettu ja ne on korvattu SignaturePurpose-jäsenmuuttujalla. Kenttä on String-tyyppiä ja sille tehdään ohjelmatasolla saniteettitarkastelu; kenttä voi saada arvokseen vain [PreparedBy | ControlledBy | ApprovedBy]. Nämä uudet arvot kuvaavat tarkemmin tehtyä operaatiota. Tietomalliin tehdyt muutokset ovat kuvassa 14.



Kuva 14: Korjattu tietomalli. Kantaluokkaan on lisätty allekirjoituksen toistamisen kannalta tärkeää tietoa. Periytettyjä luokkia on yhtenäistetty ja lisätty puuttuvaa tietoa.

Tietomallia muuttaessa on huomioitava, että käytössä olevat allekirjoitukset eivät saa kadota kannasta. Tätä varten kirjoitettiin migraatio SQL:llä. Kirjoitettu migraatio on nähtävissä liitteessä B: SQL-skripti tietokantaskeeman päivittämiseen.

6.3 Allekirjoituksien sisällön täydentäminen

Jokaisen allekirjoituksen sisältöön on tullut muutoksia. Muutokset ovat seurausta vaatimusten ja asiakkaan käyttämän dokumentaation tarkemmasta ja syvällisemmästä ymmärtämisestä. Kaikki allekirjoitettava tieto on nyt XML-formaatissa muuttujassa SignedXML. Siirtymällä allekirjoitettaviin XML-merkkijonoihin helpotetaan tulevaisuudessa tuleviin muutoksiin reagoimista. C#-olioista on helppo luoda XML-merkkijonoja allekirjoitettavaksi. Jokaiseen allekirjoitettavaan XML-merkkijonoon on sisällytetty allekirjoittajan nimi ja allekirjoituksen kellonaika.

Merkkiainetuotantoprosessin allekirjoituksien (Sample-luokka) allekirjoitettava sisältö on muutettu listauksen 6.3 mukaiseksi XML-merkkijonoksi.

```
<?xml version="1.0" encoding="utf-16"?>
<SampleSignature xmlns:DataForSignature="PETERP/LIMS/DigitalSignature">
  <BatchCode>15FDG043AV</BatchCode>
  <Status>SynthesisOperator</Status>
  <SignerName>CN=Vuorenpää Tes Aili 01098700873, SN=Vuorenpää Tes, G=Aili, SE-
RIALNUMBER=01098700873, T="100 sairaanhoitaja, sjukskötare", C=FI</SignerName>
  <Time>2015-10-16 09:19:07Z</Time>
</SampleSignature>
```

Listaus 6.3: Merkkiainetuotantoprosessin allekirjoituksien uusi tietosisältö.

Sisältö on sama kuin ennenkin, eli allekirjoitettavaa tietoa ovat: merkkiaine-erän eräkoodi, allekirjoituspiste, allekirjoittajan nimi ja allekirjoitusaika. XML-pohjaisen toteutuksen etu on muokkaamisen helppous.

Merkkiainetilausten hyväksyntä -allekirjoitukseen (AcceptedTracerOrder-luokka) on lisätty allekirjoittajan nimi ja allekirjoitusaika. Uusi XML on listauksen 6.4 mukainen.

```
<?xml version="1.0" encoding="utf-16"?>
<TracerOrderSignatureData     xmlns:xsi="http://www.w3.org/2001/XMLSchema-in-
stance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SignerName>CN=Vuorenpää Tes Aili 01098700873, SN=Vuorenpää Tes, G=Aili, SE-
RIALNUMBER=01098700873, T="100 sairaanhoitaja, sjukskötare", C=FI</SignerName>
  <Time>21.10.2015 16:04:20</Time>
  <Orders>
    <TracerOrderSignature>
      <TracerName>[18F]FDG</TracerName>
      <Radioactivity>2607</Radioactivity>
      <OrderCode>e91a77d5-ea82-4eb4-acae-4b196e838479</OrderCode>
      <FirstInjection>18.11.2015 9:30:00</FirstInjection>
    </TracerOrderSignature>
  </Orders>
</TracerOrderSignatureData>
```

Listaus 6.4: Uusi merkkiaine-erän tuotantoon hyväksymisessä allekirjoitettava XML. Muutokset ovat punaisella.

Merkkiainetilausten hyväksyntä -allekirjoitukseen lisättiin ensimmäisen injektion aika. Aikatieto on merkkiainetilauksen perustietoa, sillä merkkiaine-erän tulee olla valmis ennen ensimmäistä injektiota.

Protokolla-allekirjoitukseen (BatchApproval-luokka) on lisätty allekirjoittajan nimi ja allekirjoitusaika. <*SubstanceTest*>-elementti on nimetty kuvaavammin ja siihen on liitetty mittausten raja-arvot, komponentin nimi ja yksikkö. Nämä ovat tietoja jotka ovat nykyisessä eräpöytäkirjassa. Uusi XML on listauksen 6.5 mukainen.

```
<?xml version="1.0" encoding="utf-16"?>
<SubstanceSignatureXml  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <substance>
    <Tracer>
      <Name>[11C]6-OH-BTA-1</Name>
      <Isotope>18F</Isotope>
      <Description>Uusi merkkiaineprotokolla</Description>
    </Tracer>
    <SubstanceTests>
        <SubstanceTestCell>
           <TestDescription>Measure Acid</TestDescription>
           <GroupCode>Reage_Ainekset</GroupCode>
           <TestCode>Acetic acid</TestCode>
           <TestCodeVersion>7</TestCodeVersion>
           <StepType>BatchInstruction</StepType>
           <StockItem>Acetic acid (glacial) 100%</StockItem>
           <ComponentName>Inst_amount</ComponentName>
           <LowerLimitValue>-1</LowerLimitValue>
```

```
            <UpperLimitValue>7</UpperLimitValue>
            <Unit>mg</Unit>
        </SubstanceTestCell>
      </SubstanceTests>
      <QAReleaseRequired>true</QAReleaseRequired>
      <Description>Testi</Description>
      <SubstanceCode>ATESTI</SubstanceCode>
    </substance>
    <SignaturePurpose>ControlledBy</SignaturePurpose>
    <SignerName>CN=Vuorenpää Tes Aili 01098700873, SN=Vuorenpää Tes, G=Aili, SE-
RIALNUMBER=01098700873, T="100 sairaanhoitaja, sjukskötare", C=FI</SignerName>
    <SignTime>21.10.2015 14:36:00</SignTime>
</SubstanceSignatureXml>
```

Listaus 6.5: Uusi protokollien hyväksymisessä allekirjoitettava XML. Muutokset ovat punaisella.

Korjattu allekirjoitus sisältää *<SignaturePurpose>*-elementin. Elementti sisältää allekirjoituksen tarkoituksen. Allekirjoituksen tarkoitukset ovat [PreparedBy | ControlledBy | ApprovedBy].

6.4 Muutoksien testaus

Muutokset testataan olemassa olevilla allekirjoituksilla ja korjatulla toteutuksella. Testaus tapahtuu seuraavalla tavalla:

1. Otetaan kopio tuotantokäytössä olevasta allekirjoitustietokannasta ja otetaan tuotantokannan kopio käyttöön kehitysympäristössä.
2. Ajetaan liitteessä B listattu tietokantamigraatio. Migraatiossa poistetaan vanhat LIMS-allekirjoitukset, sillä LIMS ei ole tuotantokäytössä. PETERP-allekirjoitukset jäävät tietokantaan.
3. Luodaan allekirjoituksia käyttämällä korjattua toteutusta. Jokaista allekirjoitusta tulee olla vähintään yksi kappale.
4. Tarkistetaan, että jokainen allekirjoitus on vahvistettavissa, ja että PETERP-allekirjoituksia ei ole kadonnut kannasta.

Ensimmäinen kohta on helposti toteutettavissa. Käytössä oleva Microsoftin SQL Server mahdollistaa varmuuskopioiden ottamisen. Tuotantotietokanta on kooltaan vain muutamia kymmeniä megatavuja, joten kyseessä on nopea operaatio. Tuotantokannasta otetun varmuuskopion palauttaminen kehitysympäristöön kestää muutamia sekunteja.

Toisessa kohdassa ajetaan luotu tietokantamigraatio kehitysympäristössä käyttöön otettuun tietokantaan. Tietokantamigraatio ei saa antaa virheitä ja sen on mentävä kerralla läpi.

Kolmas kohta vaatii PETERP-asiakasohjelman, palvelimen ja Matrix Gemini LIMS-ohjelmiston korjatut versiot. Jokaista allekirjoitusta tulee olla vähintään yksi kappale, sillä tehdyt muutokset ovat vaikuttaneet jokaiseen allekirjoitustyyppiin.

Kohdassa neljä jokainen tietokannassa oleva allekirjoitus vahvistetaan. Jokainen allekirjoitus tulee vahvistua ongelmitta. Vahvistaminen tapahtuu sitä varten luodulla palvelinkutsulla, joka käy jokaisen allekirjoituksen läpi ja vahvistaa sen. Tässä koodissa on otettava huomioon, että vanhat allekirjoitukset täytyy vahvistaa eri sarjallistamisfunktiolla.

```
private bool verifySignature(DigitalSignature sig, string xml)
{
    byte[] serializedString;
    if (sig.SerializationMethod == "MemoryStream+BinaryFormatter v1")
        serializedString = DigitalSignatureHelper.CalculateHashVersion1(xml);
    else if (sig.SerializationMethod == "UnicodeEncoding.GetBytes(String)")
        serializedString = DigitalSignatureHelper.SerializeUnicodeString(xml);
    else
        throw new FaultException<DataFault>(new DataFault()
        { Operation = "VerifySampleSignature", Message = "Unknown serialization method" });

    if (!DigitalSignatureHelper.VerifyData(serializedString, sig.Signature, sig.Key))
    {
        return false;
    }
    return true;
}
```

Kuva 15: Allekirjoituksen vahvistamisessa käytetty funktio. CalculateHashVersion1-funktio on alleviivattu vihreällä. Tämä kertoo ohjelmoijalle, että kyseistä funktiota ei tulisi käyttää enää allekirjoituksien tekemiseen.

Kuvassa 15 esitetty funktio vahvistaa yhden allekirjoituksen. Funktio sarjallistaa parametrina saamansa merkkijonon serializedString-muuttujaan. Sarjallistaminen tapahtuu joko vanhalla *CalculateHashVersion1*-funktiolla tai korjatulla *SerializeUnicodeString*-funktiolla, riippuen onko kyseessä ennen vai jälkeen korjausta luotu allekirjoitus. Toteutuksessa nähdään, että ohjelmointiympäristä varoittaa ohjelmoijaa ”Obsolete”-attribuutilla merkatun funktion käytöstä alleviivaamalla sen vihreällä. Seuraavaksi sarjallistettu tieto, allekirjoitus ja allekirjoituksessa käytetty avain annetaan funktiolle, joka vahvistaa allekirjoituksen. Tarkistuksessa käytettävä koodi on nähtävissä kokonaisuudessaan liitteessä C.

Allekirjoitusten muutokset testattiin onnistuneesti kehitysympäristössä. Korjattu toteutus asennettiin tuotantopalvelimelle versiopäivityksen yhteydessä. Kaikki tietokannassa olevat PETERP-järjestelmän allekirjoitukset vahvistettiin onnistuneesti tuotantopalvelimella.

7. JATKOKEHITYS

Luku jakautuu kahteen kokonaisuuteen: korjatun toteutuksen jatkokehittämiseen ja jatkokehitysideaan. Työssä esitelty allekirjoitusten toteutus on teknisesti riittävä ja täyttää Suomen lain sekä GMP-standardin asettamat vaatimukset. Toteutusta voi kuitenkin tulevaisuudessa kehittää eteenpäin kohti parempaa käyttäjäkokemusta ja turvallisempaa toteutusta.

7.1 Korjatun toteutuksen jatkokehitys

Korjatussa teknisessä toteutuksessa ei ole jäljellä puutteita jotka estäisivät sen tuotantokäytön. Korjattua toteutusta voi kuitenkin jatkokehittää. Jatkokehitysideat jaetaan tekniseen jatkokehittämiseen, sisällön jatkokehittämiseen ja visualisoinnin jatkokehittämiseen.

7.1.1 Teknisen toteutuksen jatkokehitys

Luvussa 5.2 esitellyistä teknisistä puutteista ei ole korjattu kohtia 2, 3 ja 7:

1. Allekirjoituksessa käytettyjä algoritmeja ei allekirjoiteta.
2. Aina ei ole selvää mitä dataa allekirjoitetaan.
3. Palvelimella allekirjoitus tarkistetaan palvelinkutsun mukana tullutta allekirjoitettua vastaan. Allekirjoitetun tiedon oikeellisuudesta ei kuitenkaan varmistuta.

Jokainen listatuista kohdista on mahdollista toteuttaa korjattuun toteutukseen. Ensimmäinen kohta on vaikeudeltaan triviaali. Toteutukseen kuluva aika tulisi olemaan muutamia tunteja. Korjauksen tuoma lisäarvo on kuitenkin minimaalinen, sillä allekirjoitus on vahvistettavissa vain oikeilla algoritmeilla.

Toinen kohta korjattiin osaksi. Korjatussa toteutuksessa allekirjoitettava tieto sijaitsee SignedXML-nimisessä jäsenmuuttujassa. Tiedon voisi kuitenkin esimerkiksi siirtää kantaluokkaan. Muutos kuitenkin vaatii koodin refaktorointia ja saavutettava etu on minimaalinen. Tästä syystä kohtaa ei korjattu kokonaisuudessaan korjattuun toteutukseen. Tässä muutoksessa on myös haittapuoli: allekirjoitettavan tiedon säilyttäminen perityssä luokassa mahdollistaa myös binääritiedostojen allekirjoittamisen. Binääritiedostojen allekirjoittaminen tapahtuisi samalla tavalla kuin merkkijonojen allekirjoittaminen. Binääritiedostojen ei tarvitse sarjallistaa tavujonoksi, sillä ne ovat valmiiksi binäärimuodossa. Tietokannan VARBINARY-luokka vastaa sisällöltään C#-kielen tavujonoa. Binääritiedoston voi antaa suoraan *RSACryptoServiceProvider*-luokan *SignData*-jäsenfunktiolle.

Kolmas kohta on korjaamatta jätetyistä puutteista merkittävin. Ongelmana on, että palvelin ei tiedä millainen oikea allekirjoitettava XML-tiedosto on. Ongelman korjauksessa allekirjoitettava XML tulee luoda palvelimella. Korjaus voidaan tehdä kahdella eri tavalla:

1. Asiakassovellus lähettää palvelimelle kutsun, jonka vastauksena palvelin tuottaa allekirjoitettavan XML-merkkijonon. Asiakas allekirjoittaa XML-merkkijonon ja lähettää sen takaisin palvelimelle. Palvelin tarkistaa, että asiakkaalle lähetetty merkkijono vastaa allekirjoitettua merkkijonoa.
2. Asiakassovellus luo allekirjoitettavan XML-merkkijonon ja lähettää allekirjoituksen palvelimelle. Palvelin katsoo DigitalSignature-olion jäsenmuuttujista allekirjoitukseen liittyvät metatiedot ja luo uuden XML-merkkijonon hakemalla siihen liittyvät tiedot LIMS-tietokannasta metatiedon perusteella. Luotua merkkijonoa verrataan asiakassovellukselta saatuun merkkijonoon. Jos ne ovat samat, on allekirjoituksen sisältö kunnossa.

Ehdotetuista vaihtoehdoista kohta 2. vaikuttaisi olevan helpommin rakennettavissa nykyisen toteutuksen päälle. XML-merkkijonon uudelleen luomiseen palvelimella voi käyttää samoja kirjastoja kuin asiakassovelluksessa.

7.1.2 Allekirjoitusten sisällön jatkokehitys

Luvussa 5.3 esitetyistä puutteista kohta 4 (Paperisessa allekirjoituksessa ”End Product” vaiheessa aktiivisuus on allekirjoitettavaa tietoa) jäi työssä korjaamatta. Allekirjoitettavan tiedon sisältö on kuitenkin sovittu yhdessä projektin asiakkaan kanssa, joten muutos ei ole tarpeellinen. Kaikkien allekirjoitusten sisältö on muutettu XML-merkkijonoksi, joten asiakasvaatimusten muuttuessa kaikkien allekirjoitusten sisältöä on korjatussa toteutuksessa helppoa muokata.

7.1.3 Visualisoinnin jatkokehitys

Korjattu versio ei parantanut allekirjoitettavan tiedon visualisointia. Jokainen luvussa 5.4 esitetty kohta on jatkokehityskohde. Korjatussa toteutuksessa kaikki allekirjoitettava tieto on sarjallistettu XML-merkkijonoiksi. XML-merkkijonon merkittävä etu on lyhyt semanttinen välimatka. XML-merkkijonon esittäminen käyttöliittymässä on tarvittaessa mahdollista ja helposti toteutettavissa. Visualisointi voidaan hoitaa esimerkiksi tallentamalla XML-merkkijono tiedostoon ja avaamalla tiedosto verkkoselaimella tai näyttämällä merkkijono allekirjoituskäyttöliittymässä.

7.2 Puhtaalta pöydältä aloittaminen

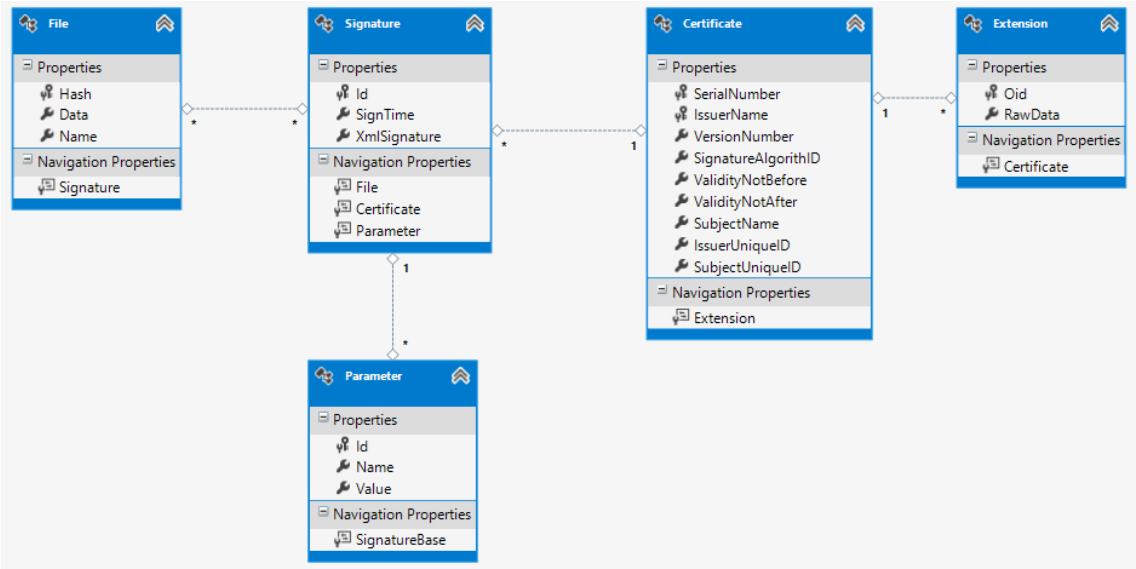
Jos allekirjoituksen toteutus tehtäisiin nyt uudelleen, olisi toteutus erilainen. Korjattu toteutus rajoittaa allekirjoitukset merkkijonoihin. On kuitenkin hyödyllistä tarjota käyttäjille mahdollisuus allekirjoittaa merkkijonojen lisäksi muutakin, kuten esimerkiksi pdf tai doc -formaatteja. Luvussa esitellään jatkokehitysideana joustavampi tietomalli ja puhutaan esitetyn jatkokehitysidean eduista visualisoinnin kannalta. Luvussa ei esitellä allekirjoituksien luomisen teknisiä yksityiskohtia, sillä aihetta on käsitelty laajasti luvuissa 3.3.1, 4.1.4, 4.1.5 ja 4.1.6.

Luvussa 3.3.2 esitelty BDOC-allekirjoitusformaatti mahdollistaa missä tahansa formaatissa olevien tiedostojen allekirjoittamisen. Allekirjoittaminen tapahtuu XML-allekirjoitusten avulla. BDOC:n käyttämä XML-allekirjoitus on omassa tiedostossaan ja se viittaa allekirjoitettuihin tiedostoihin <*reference*>-elementin avulla. Eräs XML-allekirjoitusten suurimmista eduista on allekirjoittamisen helppous. .NET-komponenttikirjasto tarjoaa XML-allekirjoituksien luomiseen valmiita työkaluja. XML-allekirjoituksen merkittävä etu on, että se sisältää kaiken tarvittavan allekirjoituksen vahvistamiseen. Allekirjoitettava data voi sijaita erillisessä tiedostossa, mikäli käytetään *detached*-tyyppistä XML-allekirjoitusta.

BDOC-formaatti ei sovellu suoraan tässä työssä esitettyyn RK-laboratorion käyttötapauksiin, sillä se on vain tarkoitettu tiedostojen allekirjoittamiseen. Digitaalisen allekirjoittamisen lisäksi tieto pitää olla tallennettavissa verkon yli palvelimelle ja tietoon pitää pystyä tekemään hakua vaihtelevilla parametreilla.

7.2.1 Jatkokehitettävä tietomalli

Jatkokehitystietomallin ideana on tallentaa jokainen allekirjoitettava tiedosto tietokantaan, käyttää XML-allekirjoituksia näiden allekirjoittamiseen ja mahdollistaa erilaisten hakuparametrien lisääminen dynaamisesti. Redundanttia tietoa vähennetään tallentamalla varmenteet tietokantaan vain kerran. Kuvassa 16 esitellään tietomalli, joka mahdollistaisi eri tyyppiä olevien tiedostojen allekirjoittamisen.



Kuva 16: Mahdollinen tietomalli jatkokehitykseen.

Mallin kriittisimmät tiedot sijaitsevat File-, Signature- ja Parameter -tauluissa. File taulun sisältö on esitetty taulukossa 11.

Taulukko 11: File-taulun sisältö ilman vierasavainviitteitä.

Sarake	Tyyppi	Kuvaus
Data	VARBINARY	Kaikesta allekirjoitettavasta tiedosta generoidaan tiedosto. Tiedosto voi olla: .xml, .doc, .pdf tai muu. Tähän sarakkeeseen tallennetaan allekirjoitettava tiedosto VARBINARY-muotoisena.
Hash	VARBINARY	Data-sarakkeesta laskettu tiiviste, joka toimii pääavaimena. Tämä tarkoittaa, että suuria tiedostoja ei tallenneta turhaan useaan kertaan. Tiivisteen voi laskea esimerkiksi SHA1-algoritmilla.
Name	NVARCHAR	Tiedoston nimi. Tiedoston nimen tulee sisältää pääte. Esimerkiksi raportti.pdf. Päätteen avulla käyttöjärjestelmä osaa avata tiedoston oikealla ohjelmalla.

File-taulussa sijaitsee allekirjoitettava tieto. Tieto on tallennettu VARBINARY-tyyppisenä. Tyyppi mahdollistaa minkä tahansa tiedoston tallentamisen. Tiedostosta laskettu tiiviste toimii pääavaimena. Tämä estää kopioiden tallentamisen samasta tiedostosta ja vähentää tarvittavaa levytilaa. On mahdollista käyttää pääavaimena myös juoksevaa numeroa, jos on tarpeellista tallentaa sama tiedosto useaan kertaa. Tiedoston nimi on for-

maattia [nimi].[tyyppi], esimerkiksi ”raportti.pdf”. Nimeä ja tiedostotyyppiä tarvitaan tiedostoa avatessa. Käyttöjärjestelmät tunnistavat tiedostojen tyypin nimen päätteestä ja päätteen avulla tiedosto on avattavissa oikealla ohjelmalla. Tauluun tallennetun tiedoston katselu vaatii:

1. Data- ja Name -kenttien lähettämisen asiakasohjelmistolle.
2. Asiakasohjelmisto tallentaa Date-kentän tavut Name-nimellä haluttuun tiedostopolkuun.
3. Asiakasohjelmisto avaa tiedoston.

Signature-taulun tallentuu allekirjoitus XML-muodossa. XML-allekirjoituksen merkittävä etu on luomisen helppous. .NET-komponenttikirjasto tarjoaa valmiin apuluokan XML-allekirjoitusten luomiseen. XML-allekirjoitukset ovat joustavia ja sopivat <*reference*>-elementtinsä ansiosta hyvin binääritiedostojen allekirjoittamiseen. Allekirjoitustaulun sisältö on esitetty taulukossa 12.

Taulukko 12: Signature-taulun sisältö ilman vierasavainviitteitä.

Sarake	Tyyppi	Kuvaus
Id	INT32	Taulun pääavain, joka on automaattisesti kasvava Integer32.
SignTime	DateTime	Palvelimella lisätty aika jolloin allekirjoitus on lisätty tietokantaan.
XmlSignature	NVARCHAR tai XML	XML-tiedosto, joka sisältää *detached*-tyyppisen XML-allekirjoituksen.

Jokainen allekirjoitus aikaleimataan palvelimella. Aikaleima asetetaan SignTime-kenttään.

Parameter-taulu sisältää hakuparametreja joilla allekirjoituksia voi etsiä. Haut kohdistuvat Name- ja Value -kenttiin. Haut tapahtuvat esimerkiksi listauksen 7.1 mukaisesti.

```
SELECT * FROM [Parameter] JOIN [Signature] ON ... WHERE Name = 'BatchCode' AND
Value = '2015-FDG-001'
```

Listaus 7.1: Esimerkki tiedon hakemisesta Parameter-taulun avulla.

Parametreja on tarkoitus pystyä lisäämään dynaamisesti. Parametritaulun sisältö on esitetty taulukossa 13.

Taulukko 13: Parameter-taulun sisältö ilman vierasavainviitteitä.

Sarake	Tyyppi	Kuvaus
Id	INT32	Taulun pääavain, joka on automaattisesti kasvava Integer32.
Name	NVARCHAR	Hakuparametrin nimi. Voi olla esimerkiksi ”Batch-Code”.
Value	NVARCHAR	Parametrin arvo. Voi olla esimerkiksi eräkoodi ”2015-FDG-001”.

Allekirjoituksia voi olla tietokannassa satoja tuhansia ja jokainen allekirjoitus voi sisältää useita parametreja. Tästä syystä on tärkeää indeksoida Parameter-taulu Name- ja Value -kenttien suhteen.

7.2.2 Jatkokehitysehdotuksen allekirjoitettavan tiedon visualisointi

Jatkokehitettäväksi ehdotettavan mallin merkittävä etu on allekirjoitettavan datan muuttaminen dokumenteiksi. Jokaisen tiedostotyypin katseluun on omat ohjelmansa; tekstitiedostoja voi katsella Notepadilla, pdf-tiedostoja Adobe Readerillä ja doc-tiedostoja Microsoft Wordillä. Ehdotetussa mallissa tiedon visualisointi jätetään käyttäjän valitsemalle ohjelmalle.

Tiedostojen avaaminen käyttäjän valitsemalla ohjelmalla ei poista luvussa 3.5 esitettyä visualisoinnin problematiikkaa, mutta ehdotettu toteutus antaa käyttäjälle mahdollisuuden visualisoida allekirjoitettava tiedosto haluamallaan tavalla. Alkuperäinen toteutus allekirjoittaa merkkijonon ja merkkijonon visualisointi vaihtelee allekirjoituspisteittäin. Jatkokehityssuunnitelman tietomallissa allekirjoitettava data on aina tiedostona käyttäjän kovalevyllä, joka mahdollistaa allekirjoitettavan tiedoston perinpohjaisemman tarkastelun.

8. YHTEENVETO

Työssä esiteltiin PETERP-projektissa tehty toteutus digitaalisten allekirjoitusten luomiseen. Työ keskittyi RK-laboratorion allekirjoituksiin, mutta muutoksia tuli sekä PETERP-järjestelmän allekirjoituksiin että RK-laboratorion allekirjoituksiin. Toteutusta tutkittiin ja varmistettiin, että allekirjoitukset ovat riittäviä lain ja säädösten, teknologian ja sisällön näkökulmasta. Lopuksi työn aikana löydetyt puutteet korjattiin.

Luvussa 4 esiteltiin allekirjoitusarkkitehtuurista olemassa oleva toteutus. Toteutus on tehty .NET-komponenttikirjaston avustuksella. Digitaaliset allekirjoitukset luotiin sarjallistamalla C#-olioita ennalta sovitussa järjestyksessä *MemoryStream* ja *BinaryFormatter* luokkien avulla. Tästä saatiin tuloksena tavujono joka annettiin allekirjoitettavaksi RSA-algoritmille. Allekirjoitukset tallennetaan Microsoft SQL server -tietokantaan.

Alkuperäisessä toteutuksessa havaittiin useita ongelmia, joista merkittävimmät olivat:

- C#-olioiden sarjallistaminen tuotti erilaisen tuloksen ennen ja jälkeen tietokantaan tallennuksen. Puute esti allekirjoituksien vahvistamisen tietokantaan tallentamisen jälkeen.
- Sarjallistamisessa, tiivistämisessä ja allekirjoittamisessa käytettyjä algoritmeja ei dokumentoitu allekirjoituksien yhteydessä. Puute hankaloittaa allekirjoitusten vahvistamista jälkeenpäin jos jotakin alkuperäisistä algoritmeista muutetaan.
- Allekirjoitettavan tiedon sisällöstä puuttui allekirjoittaneen henkilön nimi sekä kellonaika. Nämä vaatimukset tulevat GMP-standardista. Nimen ja kellonajan puute olisi voinut johtaa LIMS-järjestelmän GMP-validoinnin epäonnistumiseen.

Havaittujen puutteiden korjaukset esiteltiin luvussa 6. Sarjallistaminen korjattiin siirtymällä allekirjoittamaan XML-merkkijonoja. C#:n merkkijonot ovat helposti sarjallistettavissa käyttämällä .NET-komponenttikirjaston tarjoamaa luokkaa. Sarjallistamisessa, tiivistämisessä ja allekirjoituksessa käytetyt algoritmit jäävät korjatussa toteutuksessa tietokantaan. Kaikkien allekirjoitusten sisältöön lisättiin allekirjoittavan käyttäjän nimi ja allekirjoitusaika. Korjauksien ohessa tehtiin muita muutoksia jotka helpottavat jatkokehitystä, lisättiin puuttuvaa metatietoa ja yhtenäistettiin tietomallia. Korjaukset testattiin onnistuneesti käyttämällä liitteen C koodia. Korjattu toteutus on asennettu tuotantoympäristöön.

Lopuksi luvussa 7 esiteltiin jatkokehitysideoita. Osa esitellyistä ideoista on lisättävissä nykyiseen toteutukseen. Osa tietomallia muuttavista jatkokehitysideoista on käyttöönotettavissa tulevissa projekteissa. Työn lopputuloksena syntyi korjattu toteutus digitaalisille allekirjoitukselle. Korjattu toteutus korjaa monta alkuperäisen toteutuksen ongelmaa. Korjattu toteutus ei ole täydellinen, mutta se täyttää kaikki sille asetetut vaatimukset.

LÄHTEET

[1] Turku PET Centre, Measurement of tissue blood volume with [15O] CO PET. Saatavissa: http://www.turkupetcentre.fi/index.php?option=com_content&view=article&id=245&Itemid=104&lang=en, HTML-dokumentti. Viitattu 27.3.2015.

[2] G. B. Saha, Basics of PET Imaging, Springer Science+Business Media, 2010.

[3] IAEA, Strategies for Clinical Implementation and Quality Management of PET Tracers, 2009, 11-13 s, 129 s. Saatavissa: http://www-pub.iaea.org/MTCD/publications/PDF/pub1344_web.pdf, PDF-dokumentti. Viitattu 24.4.2015.

[4] Microsoft, Enveloped Signatures. Saatavissa: https://msdn.microsoft.com/en-us/library/ms767623%28v=vs.85%29.aspx, HTML-dokumentti. Viitattu 24.4.2015.

[5] Microsoft, Enveloping Signatures. Saatavissa: https://msdn.microsoft.com/en-us/library/ms753722%28v=vs.85%29.aspx, HTML-dokumentti. Viitattu 24.4.2015.

[6] Microsoft, Detached Signatures. Saatavissa: https://msdn.microsoft.com/en-us/library/ms759193%28v=vs.85%29.aspx, HTML-dokumentti. Viitattu 24.4.2015.

[7] D. Eastlake, J. Reagle, D. Solo, F. Hirsch, T. Roessler, XML Signature Syntax and Processing (Second Edition), 2008. Saatavissa: http://www.w3.org/TR/xmldsig-core/, HTML-dokumentti. Viitattu 24.4.2015.

[8] Sertifitseerimiskeskus, About SK. Saatavissa: https://www.sk.ee/en/about/, HTML-dokumentti. Viitattu 24.4.2015.

[9] Sertifitseerimiskeskus, BDOC file format. Saatavissa: http://www.id.ee/?id=34336, HTML-dokumentti. Viitattu 24.4.2015.

[10] Sertifitseerimiskeskus, BDOC – FORMAT FOR DIGITAL SIGNATURES, versio 2.1, 2013. Saatavissa: http://www.id.ee/public/bdoc-spec21.pdf, PDF-dokumentti. Viitattu 25.4.2015

[11] J. Boyer, Canonical XML Version 1.0, 2001. Saatavissa: http://www.w3.org/TR/xml-c14n#Examples, HTML-dokumentti. Viitattu 24.4.2015.

[12] O. Kömmerling, M. G. Kuhn, Design Principles for Tamper-Resistant Smartcard Processors, 1999. Saatavissa: https://www.usenix.org/legacy/events/smartcard99/full_papers/kommerling/kommerling.pdf, PDF-dokumentti. Viitattu 17.6.2015.

[13] Laki vahvasta sähköisestä tunnistamisesta ja sähköisistä allekirjoituksista, 7.8.2009/617. Saatavissa: http://www.finlex.fi/fi/laki/ajantasa/2009/20090617

[14] A. Jøsang, B. AlFayyadh, 2008, Robust WYSIWYS: A Method for Ensuring that What You See Is What You Sign. Saatavissa: http://folk.uio.no/josang/papers/JA2008-AISC.pdf, PDF-dokumentti. Viitattu 17.6.2015.

[15] Evira, hyvät tuotantotavat, GMP. Saatavissa: http://www.evira.fi/portal/fi/elintarvikkeet/valmistus+ja+myynti/kontaktimateriaalit/hyvat+tuotantotavat++gmp/, HTML-dokumentti. Viitattu 10.7.2015.

[16] European Commission Health and comsumers directorare-general. Good Manufacturing Practice: Medicinal Products for Human and Veterinary Use, Annex 11: Computerised Systems, Electronic Signature & Batch release, 2011. Saatavissa: http://ec.europa.eu/health/files/eudralex/vol-4/annex11_01-2011_en.pdf, PDF-dokumentti. Viitattu 23.7.2015.

[17] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, IETF, 2008. Saatavissa: https://tools.ietf.org/html/rfc5280

[18] M. Linden, Julkisen avaimen järjestelmä, toimikortit ja niiden soveltaminen organisaatiossa, 2002. Saatavissa: https://dspace.cc.tut.fi/dpub/bitstream/handle/123456789/22594/Linden.pdf?sequence=1&isAllowed=y, PDF-dokumentti. Viitattu 17.8.2015.

[19] Väestörekisterikeskus, kortinlukijaohjelmisto ja varmenteen testaus. Saatavissa: https://eevertti.vrk.fi/Default.aspx?id=247, HTML-dokumentti. Viitattu 24.9.2015.

[20] Väestörekisterikeskus, usein kysyttyä: miksi korttia sanotaan turvalliseksi? Saatavissa: http://vrkfineid.javerdel.com/default.aspx?id=338 HTML-dokumentti. Viitattu 14.1.2015.

[21] Väestörekisterikeskus, FINEID S2 - VRK (PRC) CA-model and certificate contents, v2.4, 2013. Saatavissa: https://eevertti.vrk.fi/Default.aspx?id=0&docid=951&action=Publish, PDF-dokumentti. Viitattu 14.11.2015.

[22] Microsoft, BinaryFormatter Class. Saatavissa: https://msdn.microsoft.com/en-us/library/vstudio/System.Runtime.Serialization.Formatters.Binary.BinaryFormatter%28v=vs.90%29.aspx, HTML-dokumentti. Viitattu 25.9.2015.

[23] Microsoft, DateTime.Ticks Property. Saatavissa: https://msdn.microsoft.com/en-us/library/system.datetime.ticks%28v=vs.110%29.aspx, HTML-dokumentti. Viitattu 25.9.2015.

[24] Microsoft, DateTime.Kind Property. Saatavissa: https://msdn.microsoft.com/en-us/library/system.datetime.kind%28v=vs.110%29.aspx, HTML-dokumentti. Viitattu 25.9.2015.

[25] Liikenne- ja viestintäministeriö, Vaatimusten arviointi: Turvalliset sähköisen allekirjoituksen luomisvälineet, 2004. Saatavissa: http://www.lvm.fi/fileserver/52_2004.pdf, PDF-dokumentti. Viitattu 6.10.2015.

[26] X. Wang, H. Yu, How to break MD5 and other hash functions, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques Aarhus, Denmark, May 22–26, 2005.

[27] M. Stevens, P. Karpman, T. Peyrin, Freestart collision for full SHA-1, Cryptology ePrint Archive, Report 2015/967. Saatavissa: https://eprint.iacr.org/2015/967.pdf, PDF-dokumentti. Viitattu 14.11.2915.

LIITE A: MERKKIAINEPROTOKOLLAN ERÄPÖYTÄKIRJA

TURKU PET CENTRE | MET5516/Appendix 11 Version 2 | 1(2)

BATCH PROTOCOL
Standard manufacturing procedure for [^{18}F]FTHA injection

Date		**Delivery sheet no.**	
Batch code		**PET study numbers**	
Initials			

PRE PRODUCTION PROCEDURES	**Initials**
Activation of SepPak (Plus C18 ☐ / Light C18☐); EtOH (7.0±1.0) ml + H_2O (10.0 ± 1.0) ml	
Cleaning and drying the lines of apparatus	
Cleaning and preparation of SFU	

REAGENTS				
Reaction vessel	Lot no. / Exp. date	Theor. amount	Measured	Initials
Kryptofix 2.2.2		12.0 ± 1.0 mg	mg	
CH_3CN		0.10 ± 0.05 ml	ml	
Precursor solution	Lot no. / Exp. date	Theor. amount	Measured	Initials
Precursor		4.0 ± 0.5 mg	mg	
CH_3CN		1.0 ± 0.1 ml	ml	
Reagent mixtures	Lot no. / Exp. date	Theor. amount	Measured	Initials
Mixture A: AcOH +		166 ± 10 µl	µl	
preparat. HPLC-eluent		600 ± 10 µl	µl	
Mixture B: 0,1 M Phosphate buffer +		60.0 ± 0.5 ml	as is	
Ascorbic acid		120.0 ± 10.0 µl	µl	
Other	Lot no. / Exp. date	Theor. amount	Measured	Initials
2 M KOH		1.4 ± 0.1ml	ml	
EtOH		1.0 ± 0.1 ml	ml	

SYNTHESIS DATA				**Initials**
Target water removal	°C	1. min	2. min	
Radiofluorination	°C	-	min	
Drying of CH_3CN	He bar	-	min	
Hydrolysis	°C	-	min	

SEMI-PREPARATIVE HPLC				**Initials**
Column: Phenomenex Gemini 10 µ C18 110A	Lot			
Eluent: MeOH/H_2O/AcOH/Ascorbic acid (5 days)	Lot	Exp.date		
Flow	-		ml/min	
FOR FORMULATION	Lot no. / Exp. date	Theor. amount	Measured	Initials
Syringe CS05	Mixture B	10.5 ± 0.5 ml	ml	
Syringe CS06	Mixture B	20.0 ± 2.0 ml	ml	
Syringe CS07	Mixture B	20.0 ± 2.0 ml	ml	
Sterile vial unit		-	-	

END PRODUCT			Remarks
Volume	9.0-11.0 ml	ml	
Sterile filter PHT	RPD < 10 %	%	
Activity		GBq	
Time		:	
Appearance	Clear + colorless solution	Pass / Failed	
Synthesis made by			

QUALITY CONTROL FOR [^{18}F]FTHA

Analytical HPLC			Initials
Column: Phenomenex Gemini 10 µ C18 110A	Lot		
Eluent A: MeOH/AcOH (3 mo)	Lot	Exp. date	
Eluent B: H_2O/AcOH (1 w)	Lot	Exp. date	
Blank solution (3 mo)	Lot	Exp. date	
Injected volume		µl	
FTHA reference solution (2 w)	Lot	Exp. date	
Injected volume		µl	
Injected concentration		mg/ml	
Injected amount		mg	
R_t FTHA (R_t1)		min	

QUALITY CONTROL TESTS BEFORE RELEASING THE PRODUCT FOR USE

Test solution	Theoretical value	Measured	Initials
Injected volume	-	µl	
R_t [^{18}F]FTHA (R_t2)	-	min	
Identity	$R_t2 = R_t1 \pm 1.0$ min	Pass / Failed	
Content of FTHA	≤ 0.5 mg/ml	Pass / Failed	
Chemical impurities / compound	≤ 0.5 mg/ml	Pass / Failed	
Radiochem. purity	≥ 95.0 %	%	
pH	5-8	Pass / Failed	

Kryptofix 2.2.2			Initials
Reference solution (3 mo)	Lot	Exp. date	
Concentration		mg/ml	
Test solution	< 0.22 mg/ml	Pass / Failed	

Remarks	Copy of the Label
	Initials:
	Label checked:

	Signature	Date	Time
Batch released/rejected by			
QA approved/rejected by			-

LIITE B: SQL-SKRIPTI TIETOKANTASKEEMAN PÄIVITTÄMISEEN

```
--Pitää huolen että komennot menevät PetErpDigitalSignature-tietokantaan
USE PetErpDigitalSignature;
GO

--Kantaluokan muutokset-----------------------------------------
--Lisätään uudet rivit
ALTER TABLE DigitalSignature ADD MessageDigest NVARCHAR(200) NULL;
ALTER TABLE DigitalSignature ADD SignatureAlgorithm NVARCHAR(200) NULL;
ALTER TABLE DigitalSignature ADD SerializationMethod NVARCHAR(500) NULL;
GO

--Päivitetään vanhoille sarakkeille käytetyt arvot: Käytetty MD5 ja RSA.
UPDATE DigitalSignature SET MessageDigest = 'MD5+SHA1CryptoServiceProvider';
UPDATE DigitalSignature SET SignatureAlgorithm = 'RSACryptoServiceProvider';
UPDATE DigitalSignature SET SerializationMethod = 'MemoryStream+BinaryFormatter
v1';

GO

--Muutetaan sarakkeet ei-nullable, sillä kyseessä on pakollinen tieto.
ALTER TABLE DigitalSignature ALTER COLUMN MessageDigest NVARCHAR(200)  NOT NULL;
ALTER TABLE DigitalSignature ALTER COLUMN SignatureAlgorithm NVARCHAR(200) NOT
NULL;
ALTER TABLE DigitalSignature ALTER COLUMN SerializationMethod NVARCHAR(500) NOT
NULL;

GO



--Kantaluokan muutokset loppuu ----------------------------------

--Poista LIMS allekirjoitukset; eivät ole tuotannossa.
DELETE FROM DigitalSignature WHERE Id IN (SELECT id FROM DigitalSignature_Sam-
ple);
DELETE FROM DigitalSignature_Sample;
GO
DELETE FROM DigitalSignature WHERE Id IN (SELECT id FROM DigitalSignature_Accept-
edTraceOrder);
DELETE FROM DigitalSignature_AcceptedTraceOrder;
GO
DELETE FROM DigitalSignature WHERE Id IN (SELECT id FROM DigitalSigna-
ture_BatchApproval);
DELETE FROM DigitalSignature_BatchApproval;
DELETE FROM DigitalSignature_Acceptor;
DELETE FROM DigitalSignature_Compiler;
DELETE FROM DigitalSignature_Inspector;

--Poista turhat taulut
IF OBJECT_ID(N'[dbo].[FK_Acceptor_inherits_BatchApproval]', 'F') IS NOT NULL
    ALTER TABLE [dbo].[DigitalSignature_Acceptor] DROP CONSTRAINT [FK_Accep-
tor_inherits_BatchApproval];
GO
IF OBJECT_ID(N'[dbo].[DigitalSignature_Acceptor]', 'U') IS NOT NULL
    DROP TABLE [dbo].[DigitalSignature_Acceptor];
GO
```

```
IF OBJECT_ID(N'[dbo].[FK_Compiler_inherits_BatchApproval]', 'F') IS NOT NULL
    ALTER TABLE [dbo].[DigitalSignature_Compiler] DROP CONSTRAINT [FK_Com-
piler_inherits_BatchApproval];
GO
IF OBJECT_ID(N'[dbo].[DigitalSignature_Compiler]', 'U') IS NOT NULL
    DROP TABLE [dbo].[DigitalSignature_Compiler];
GO

IF OBJECT_ID(N'[dbo].[FK_Inspector_inherits_BatchApproval]', 'F') IS NOT NULL
    ALTER TABLE [dbo].[DigitalSignature_Inspector] DROP CONSTRAINT [FK_Inspec-
tor_inherits_BatchApproval];
GO
IF OBJECT_ID(N'[dbo].[DigitalSignature_Inspector]', 'U') IS NOT NULL
    DROP TABLE [dbo].[DigitalSignature_Inspector];
GO

--AcceptedTraceOrder muutokset alkaa------------------------------
--Uudelleen nimetään muutettu sarake käyttämällä sisäänrakennettua funktiota.
EXEC sp_RENAME 'DigitalSignature_AcceptedTraceOrder.XMLSerializedData' ,
'SignedXML', 'COLUMN';
GO

--Poistetaaa; tieto sisällytetään allekirjoitettavaan XML:n ja Date tieto löytyy
kantaluokasta
ALTER TABLE DigitalSignature_AcceptedTraceOrder DROP COLUMN SignDate;
GO

-- Creating table 'AcceptedBatchSet'
CREATE TABLE [dbo].[AcceptedBatchSet] (
    [Id] int IDENTITY(1,1) NOT NULL,
    [BatchGuid] nvarchar(max)  NOT NULL,
    [AcceptedBatchAcceptedTraceOrder_AcceptedBatch_Id] int  NOT NULL
);
GO

-- Creating primary key on [Id] in table 'AcceptedBatchSet'
ALTER TABLE [dbo].[AcceptedBatchSet]
ADD CONSTRAINT [PK_AcceptedBatchSet]
    PRIMARY KEY CLUSTERED ([Id] ASC);
GO

-- Creating foreign key on [AcceptedBatchAcceptedTraceOrder_AcceptedBatch_Id] in
table 'AcceptedBatchSet'
ALTER TABLE [dbo].[AcceptedBatchSet]
ADD CONSTRAINT [FK_AcceptedBatchAcceptedTraceOrder]
    FOREIGN KEY ([AcceptedBatchAcceptedTraceOrder_AcceptedBatch_Id])
    REFERENCES [dbo].[DigitalSignature_AcceptedTraceOrder]
        ([Id])
    ON DELETE NO ACTION ON UPDATE NO ACTION;

-- Creating non-clustered index for FOREIGN KEY 'FK_AcceptedBatchAcceptedTraceOr-
der'
CREATE INDEX [IX_FK_AcceptedBatchAcceptedTraceOrder]
ON [dbo].[AcceptedBatchSet]
    ([AcceptedBatchAcceptedTraceOrder_AcceptedBatch_Id]);
GO


--AcceptedTraceOrder loppuu----------------------------------------

--Sample muutokset-------------------------------------------------
```

```
ALTER TABLE DigitalSignature_Sample ADD SignedXML NVARCHAR(max) NOT NULL;
ALTER TABLE DigitalSignature_Sample DROP COLUMN SignDate;
GO
--Sample muutokset loppuu----------------------------------------

--batchApproval muutokset----------------------------------------
ALTER TABLE DigitalSignature_BatchApproval ADD SignaturePurpose NVARCHAR(max) NOT
NULL
      CHECK(SignaturePurpose = 'PreparedBy' OR SignaturePurpose = 'ControlledBy'
OR SignaturePurpose = 'ApprovedBy');
GO
--batchApproval loppuu----------------------------------------
```

LIITE C: TIETOMALLIN MIGRAATION TESTAAVA OHJELMAKOODI

```
public List<int> VerifySampleSignatures()
{
      List<int> failedId = new List<int>();
      using (var db = new
      DigitalSignatureDataModel.DigitalSignatureModelContainer())
      {
          var samples = db.DigitalSignature.
              OfType<Sample>().ToList();
          foreach (var sig in samples)
          {
              if (verifySignature(sig, sig.SignedXML) == false)
                  failedId.Add(sig.Id);
          }

          var PatientSignature = db.DigitalSignature.
              OfType<PatientSignature>().ToList();
          foreach (var sig in PatientSignature)
          {
              if (verifySignature(sig, sig.SignedXML) == false)
                  failedId.Add(sig.Id);
          }
          var CardSignature = db.DigitalSignature.
              OfType<CardSignature>().ToList();
          foreach (var sig in CardSignature)
          {
              if (verifySignature(sig, sig.SignedXML) == false)
                  failedId.Add(sig.Id);
          }
          var ScanningSignature = db.DigitalSignature.
              OfType<ScanningSignature>().ToList();
          foreach (var sig in ScanningSignature)
          {
              if (verifySignature(sig, sig.SignedXML) == false)
                  failedId.Add(sig.Id);
          }
          var AcceptedTraceOrder = db.DigitalSignature.
              OfType<AcceptedTraceOrder>().ToList();
          foreach (var sig in AcceptedTraceOrder)
          {
              if (verifySignature(sig, sig.SignedXML) == false)
                  failedId.Add(sig.Id);
          }
          var BatchApproval = db.DigitalSignature.
              OfType<BatchApproval>().ToList();
          foreach (var sig in BatchApproval)
          {
              if (verifySignature(sig, sig.SignedXML) == false)
                  failedId.Add(sig.Id);
          }
            //if (failedId.Count > 0)
          //    throw new NoGraduationException("You failed");
          return failedId;

      }
  }
```

```
private bool verifySignature(DigitalSignature sig, string xml)
{
      byte[] serializedString;
      if (sig.SerializationMethod == "MemoryStream+BinaryFormatter v1")
                    serializedString = DigitalSignatureHelper.
                    CalculateHashVersion1(xml);
      else if (sig.SerializationMethod
      == "UnicodeEncoding.GetBytes(String)")
          serializedString = DigitalSignatureHelper.
            SerializeUnicodeString(xml);
      else
          throw new FaultException<DataFault>(new DataFault()
          {
            Operation = "VerifySampleSignature", Message = "Unknown se-
            rialization method"
          });

      if (!DigitalSignatureHelper.VerifyData(serializedString,
      sig.Signature, sig.Key))
      {
          return false;
      }
      return true;
}
```